

64'er
SONDERHEFT
TIPS, TRICKS & TOOLS
SONDERHEFT 43
ÖS 100,-/Sfr. 14,-
Lit. 14000/hfl. 18,-/dkr. 72,-
DM 14,-
Markt&Technik
64'er
TIPS
TOOLS
TRICKS
Ausgewählt
Tips & Tools für jeden Zweck
■ Sprites ohne Grenzen
■ Jetzt neue Level für das Super-Spiel »Crillion«
30-Seiten-Kurs
Basic für Aufsteiger
Workshop
Giga-Publish leichtgemacht
■ Schritt für Schritt zum DTP-Profi
■ Das Ende der Druckerprobleme
ALLE PROGRAMME AUCH AUF DISKETTE ERHÄLTLICH
Knobel-Ecke
■ Auflösung: Die Sieger der ersten Aufgabe
■ Neue Herausforderung


POWER GAMES
64er online
64er-online.de
64er-online.net

REIF FÜRS MUSEUM?

Die Entwicklung immer schnellerer, leistungsfähigerer und preiswerterer Computer ist nicht aufzuhalten. Personal Computer stoßen heute in Preisbereiche vor, die früher den Homecomputern vorbehalten waren. Wird der C64 damit bald nur noch in den Ausstellungsräumen der Museen in aller Welt zu finden sein?

Die aktuellen Verkaufszahlen sagen etwas anderes: Nach wie vor zählt der C64 zu den Spitzenreitern in der Gunst des Käufers. Ein wesentlicher Grund für seine Beliebtheit ist das konkurrenzlose Preis-/Leistungsverhältnis.

Wie aktuell, konkurrenz- und wandlungsfähig der C64 heute ist, zeigen die beiden folgenden Beispiele:

- Mit keinem anderen Computer ist der Einstieg in moderne Formen der Telekommunikation wie Bildschirmtext (Btx) oder Datenfernübertragung (DFÜ) so preiswert wie mit dem C64.
- Desktop Publishing (DTP), bisher eine Domäne für einen schnellen AT oder Apple Macintosh, ist für den C64 längst kein Fremdwort mehr. Unser Workshop zum DTP-Programm Giga-Publish beweist die Leistungsfähigkeit des C64.

Wie intensiv sich unsere Leser mit dem C64 beschäftigen, zeigen die vielen Tips & Tricks, die uns jeden Tag erreichen. Eine Zusammenstellung der besten Tips finden Sie in diesem Sonderheft.

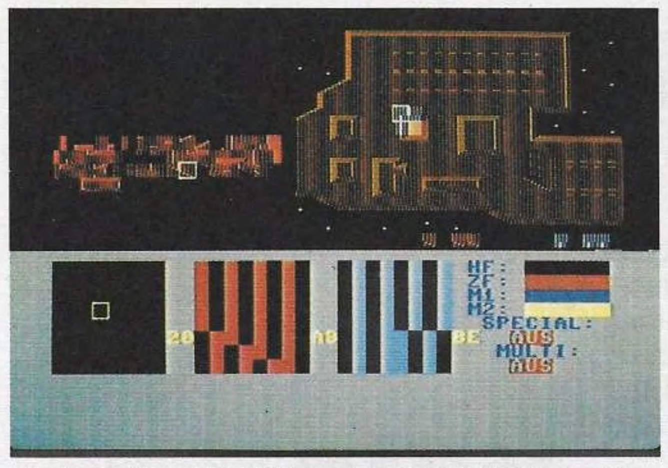
Egal, welche neuen Entwicklungen die Zukunft bringen wird – das Museum muß warten...

Ihr Elmar Friebe
(Redakteur)





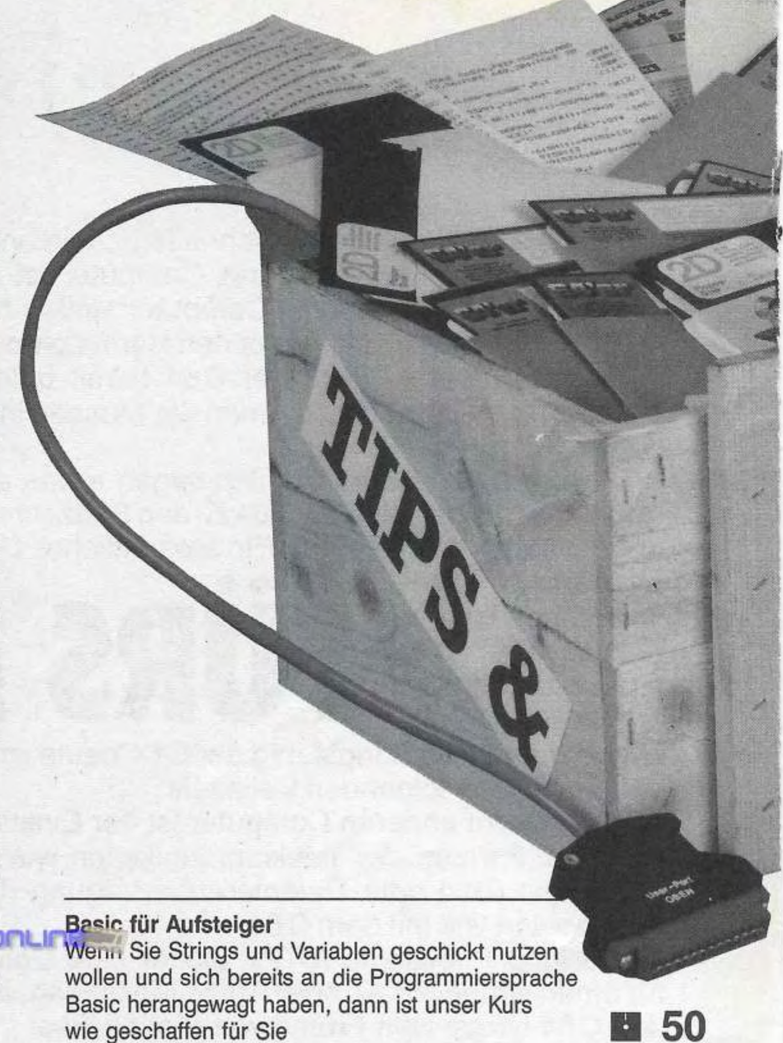
INHALT



Mit dem »Charakter-Editor« erzeugen Sie interessante Hintergrundgrafiken für eigene Spiele. Seite 16



Wenn Ihnen die 25 Level von »Crillion« nicht mehr ausreichen: Der Leveleditor sorgt für immer neue Herausforderungen. Seite 149



Grafik

Kunstwerke mit dem Computer
Der Bildschirm als Kaleidoskop: Auf fraktaler Basis entstehen wahre Kunstwerke mit unterschiedlichsten Formen 6

Universeller Zeichensatz-Editor
Mit dem »Character-Editor« entwickeln Sie mühelos ein- oder mehrfarbige Zeichensätze oder Hintergrundgrafiken für Spiele 16

Workshop

Giga-Publish
Unser Workshop zu »Giga-Publish« begleitet Sie Schritt für Schritt auf dem Weg zum DTP-Profi 30

Centronics-Treiber für Giga-Publish
Der neue Software-Treiber mit einem Centronics/ User-Port-Kabel bereitet vielen Druckproblemen bei »Giga-Publish« ein Ende 41

Kurse

Der Weg zum richtigen Ton
Lernen Sie alle Grundlagen kennen, mit deren Hilfe Sie tolle Soundeffekte programmieren können 42

Basic für Aufsteiger
Wenn Sie Strings und Variablen geschickt nutzen wollen und sich bereits an die Programmiersprache Basic herangewagt haben, dann ist unser Kurs wie geschaffen für Sie 50

Tools

Menügesteuert laden
Erstellen Sie Ihr Disketten-Menü selbst 87

Provic 64
Darstellung von 32 Sprites in vier Bildschirmbereichen – mit »Provic 64« kein Problem 90

Maskengenerator für Verwöhnte
Das Gestalten von Bildschirmmasken wird mit diesem Editierprogramm zum Kinderspiel 94

Knobelecke

Auflösung der Knobelecke 1
Die Gewinner der ersten Knobelecke aus dem Sonderheft 40: Die pfiffigen Lösungswege werden ausführlich beschrieben. 99

Knobelecke 2
Stellen Sie sich der neuen Herausforderung? Attraktive Preise warten wieder auf die Sieger. 103

Tips & Tricks

Rasterinterrupt – nicht nur für Profis
Wie die Programmierung von Raster-Interrupts funktioniert, erfahren Sie anhand eines ausführlich dokumentierten Beispielprogramms 104

Das Schatzkästchen
Zusammenstellung der besten Tips & Tricks, die das Programmieren erleichtern 120

Spielen ohne Ende
Verzweifeln Sie nicht an den Spielen aus den Sonderheften 37 oder 42: Hier finden Sie die POKEs, mit denen Sie jedes Spiel meistern. 146

Spiele-Tip

Crillion – der unendliche Spielespaß
Mit dem komfortablen Editor lassen sich die 25 Level von »Crillion« jederzeit neu gestalten. Abwechslung ist für viele Stunden garantiert 149

Eingabehilfen

Checksummer V3 und MSE
Diesen Artikel sollten Sie unbedingt lesen, wenn Sie Programme aus diesem Sonderheft abtippen wollen 159

Sonstiges

Editorial 3

Mitmachkarte 147

Vorschau 162

Impressum 162

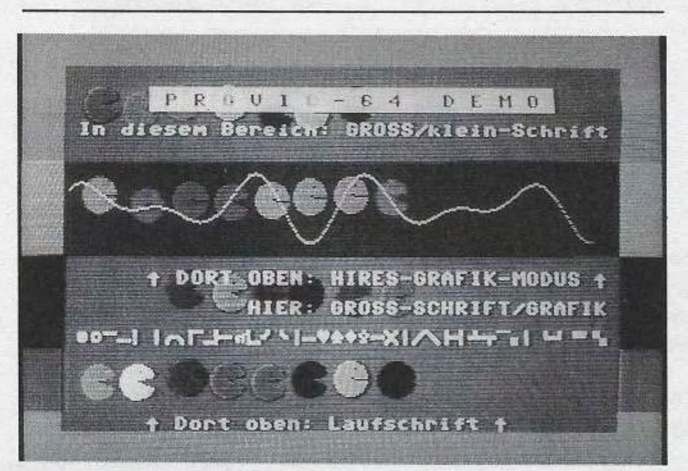
Alle Programme aus Artikeln mit einem ■ -Symbol finden Sie auch auf der Programmservice-Diskette zu diesem Sonderheft

Greifen Sie zu: Das Schatzkästchen ist mit den besten Tips und Tricks zu jedem Zweck gefüllt. Seite 120

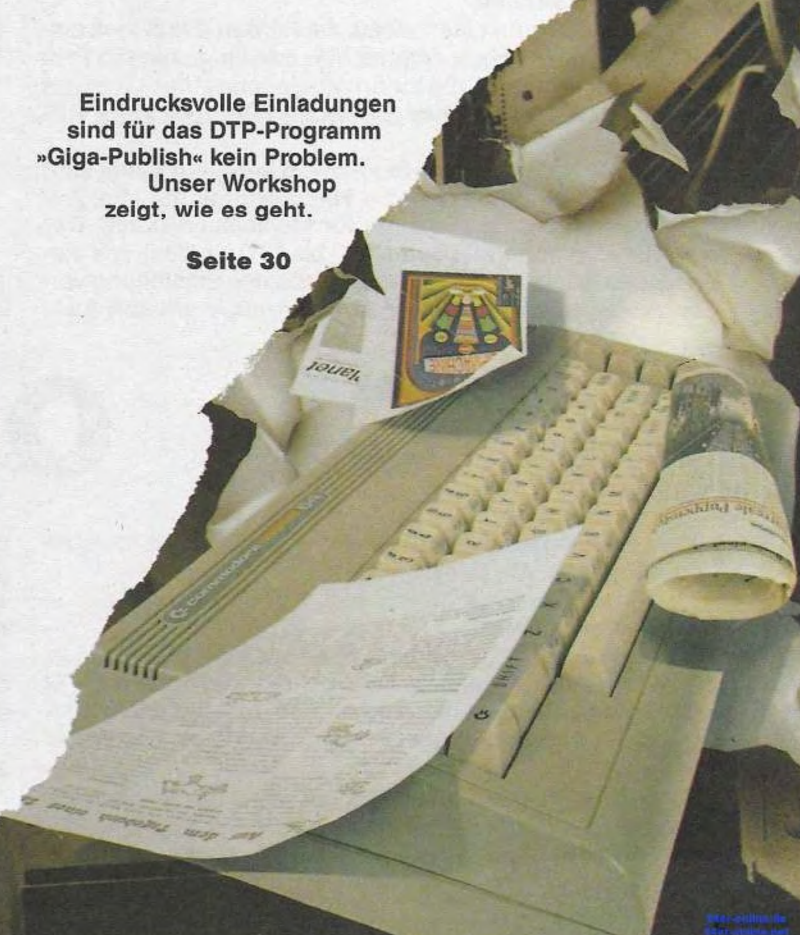
Das erste Lebenszeichen
Wissen Sie, was in Ihrem Computer nach dem Einschalten vor sich geht? 113

Tips & Tricks für Basic-Programmierer
Gestalten Sie Ihre Basic-Programme schneller und effektiver 114

Sprites ohne Grenzen
Den gesamten Bildschirm können Sie mit »Hyperscreen« für die Darstellung von Sprites und Laufschriften nutzen 119



32 Sprites gleichzeitig – nutzen Sie den gesamten Bildschirm mit »Provic 64«. Seite 90

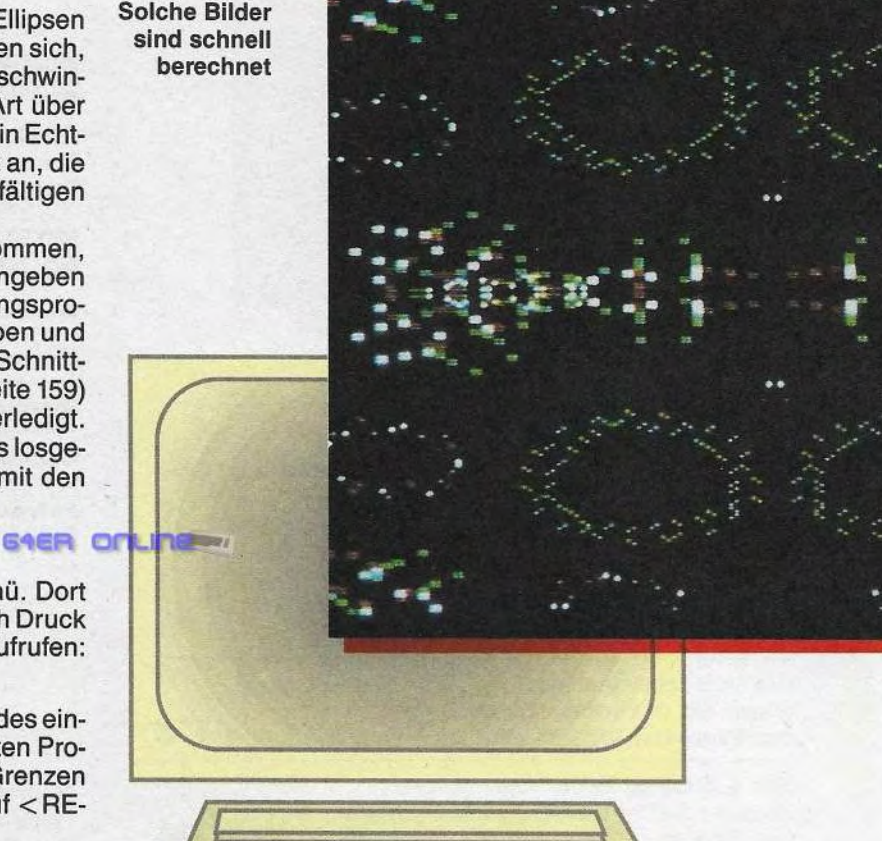


Eindrucksvolle Einladungen sind für das DTP-Programm »Giga-Publish« kein Problem. Unser Workshop zeigt, wie es geht.

Seite 30

K U N S T mit d

Nach der Pop-Art nun die Computer-Art. Unser Programm erzeugt ansprechende Grafiken auf fraktaler Basis. Anhand einfacher mathematischer Vorschriften werden flüssig ineinander übergehende Bilder mit variablem Aussehen erzeugt. Lassen auch Sie sich in ihren Bann ziehen.


Bild 1. Solche Bilder sind schnell berechnet

Das müssen Sie erlebt haben: Verschiedene Ellipsen bewegen sich wie in einem Kaleidoskop, drehen sich, wandern, neue Figuren kommen dazu, alte verschwinden wieder, geheimnisvoll wandern Formen aller Art über den Bildschirm Ihres C 64 oder C 128 – und das alles in Echtzeit! Schauen Sie sich nur einmal die Bilder 1 und 2 an, die eine (leider nicht bewegte) kleine Kostprobe der vielfältigen Möglichkeiten bietet.

Bevor Sie in den Genuß der »Computer-Art« kommen, müssen Sie allerdings zwei kleinere Programme eingeben und speichern. Listing 1 ist das Basic-Steuerungsprogramm, das Sie bitte mit dem Checksummer eingeben und unter dem Namen »C-ART« speichern. Es stellt die Schnittstelle zu Listing 2 (mit dem MSE eingeben, siehe Seite 159) her, das die Schwerstarbeit der Grafikerzeugung erledigt.

Wenn Sie beide Listings abgetippt haben, kann es losgehen. Laden und starten Sie das Basic-Programm mit den Befehlen

```
LOAD "C-ART",8,0      <RETURN>
RUN                   <RETURN>
```


Nach dem Start befinden Sie sich im Hauptmenü. Dort stehen sechs Funktionen zur Auswahl, die Sie durch Druck auf die entsprechende Zifferntaste <1> bis <6> aufrufen:

1. Grafik-Parameter:

Dieser Menüpunkt dient dazu, die Farben des Bildes einzustellen. Der Computer nimmt hier, wie im gesamten Programm, nur Werte an, die innerhalb der erlaubten Grenzen liegen. Eingaben schließen Sie jeweils mit Druck auf <RETURN> ab.

Folgende Farben stellen Sie hier ein (erlaubte Werte sind die von 0 bis 15): Rahmenfarbe, Hintergrundfarbe, drei Zeichenfarben (Farbe 2 und 3 nur für Multicolor-Modus). Der Wert von »Compound« (Bereich: 0 bis 127) legt fest, wie viele Figuren fixiert, also unabhängig von der Einstellung von »Bildwiederholung« (siehe unten, Prozeßparameter) (nur) genau einmal gezeichnet werden sollen. Die Voreinstellung für diesen Wert ist 0.

C O M P

2. Rechenparameter:

Dies sind die eigentlichen Steuerparameter, die den größten Einfluß auf das Aussehen der Figuren ausüben. »X-Fract« darf Werte von 1 bis 7 annehmen und entspricht dem X-Radius (waagerecht) der Bilder. Analog ist »Y-Fract« der Radius in Y-Richtung (senkrecht). Als Radius ist in beiden Fällen der Wert 2 voreingestellt. »X-Vorzeichen« und »Y-Vorzeichen« legt die Feinstruktur des Bildes fest. Die Voreinstellung 1 liegt innerhalb der erlaubten Grenzen 1 bis 3. Die »Krümmung« schließlich nimmt entweder den voreingestellten Wert 1 oder die Null an und bestimmt die Gesamtcharakteristik des Bildes. Falls der Wert 0 gewählt wird, müssen sich X-Fract und Y-Fract unterscheiden.

Genauere Erklärungen zu diesen Parametern würden mathematisch zu weit führen. Es ist viel sinnvoller, einfach einmal die Wirkung an einigen Bildern auszuprobieren.

3. Prozeßparameter:

Diese Einstellungen steuern den Rahmenablauf. Die Werte legen dabei den Ort und die Häufigkeit des Auftretens der Figuren auf dem Bildschirm fest.

W E R K E

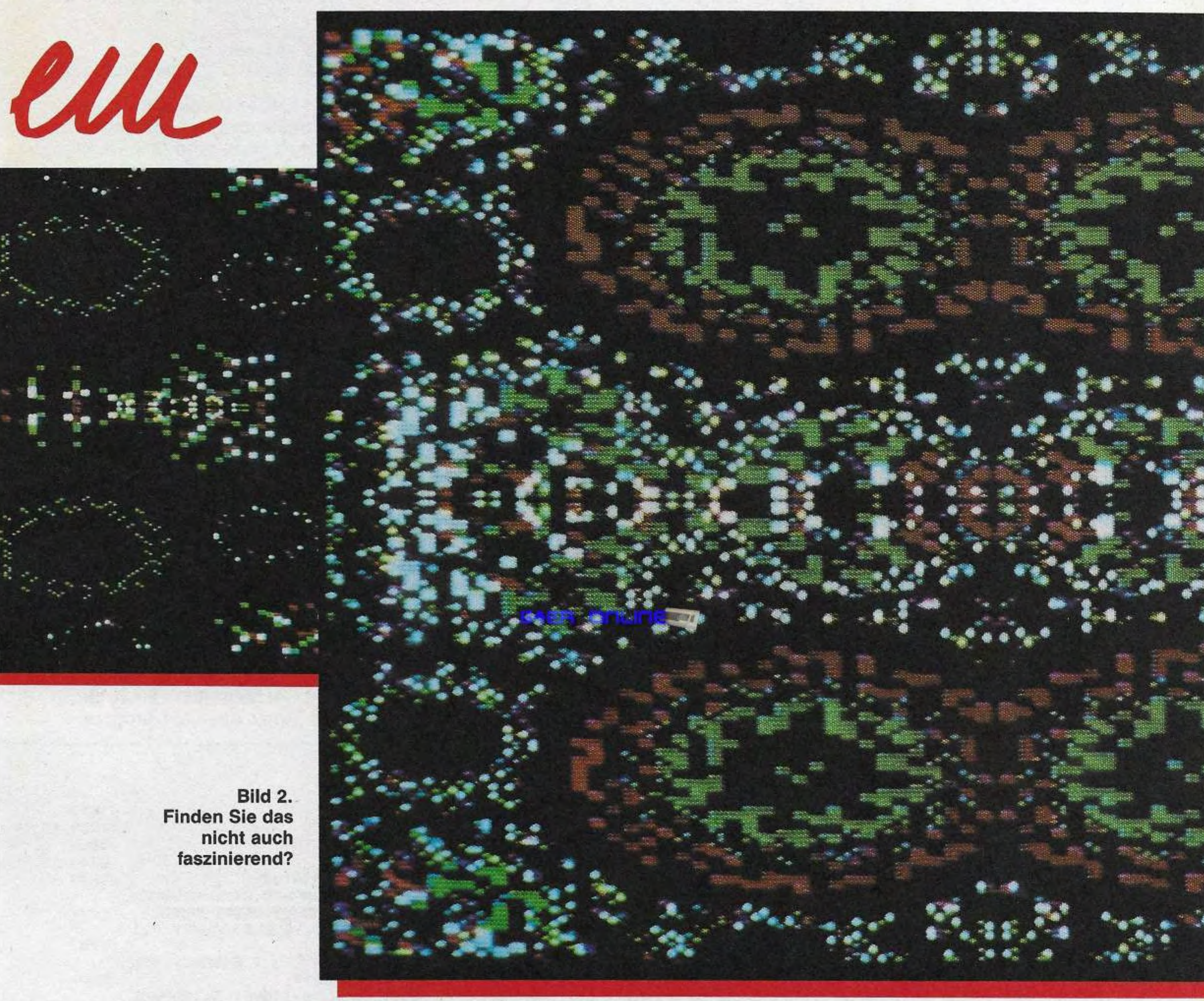


Bild 2. Finden Sie das nicht auch faszinierend?

U T E R

Für »Bildwiederholung« legen Sie fest, wie oft jede einzelne Figur gezeichnet werden soll. Beim ersten Zeichenvorgang wird die Figur erzeugt, beim zweiten Vorgang wieder gelöscht und so weiter. Daher bleibt jede Figur auf dem Schirm stehen, wenn Sie hier eine ungerade Zahl eingeben. Der erlaubte Bereich erstreckt sich von 1 bis 255, voreingestellt ist 3.

Beachten Sie, daß eine bestimmte Anzahl von Grafiken am Anfang, die mit »Compound« festgelegt wird, grundsätzlich nur einmal gezeichnet und nicht gelöscht wird.

Die »Farbstetigkeit« legt (nur im Multicolor-Betrieb) fest, nach wie vielen Punkten die Zeichenfarbe gewechselt werden soll. So werden nacheinander die drei Zeichenfarben durchgespielt, danach beginnt der Computer wieder bei der ersten. Die Farbstetigkeit liegt zwischen 1 und 255, Defaultwert ist 1. »Bildpause« gibt an, wie lange ein Motiv mindestens auf dem Schirm zu sehen sein soll, Zeitbasis ist hier etwa $^1/_{120}$ Sekunde. Der Wert liegt zwischen 1 und 255, Voreinstellung ist 18.

Die »X-Grenze« bestimmt die Größe des Zeichenfensters in X-Richtung und nimmt Werte zwischen 1 und 320 an (Voreinstellung 160). Gleiches gilt für die »Y-Grenze« (1 bis 200, Voreinstellung 100). Der Bereich der beiden »Spiegelachsen« ist derselbe wie der bei der jeweiligen Grenze, die

Achsen sind auf die Maximalwerte eingestellt. Sie bestimmen, an welchen Achsen die Punkte gespiegelt werden sollen, um den »Kaleidoskop-Effekt« zu erzeugen. Es muß der doppelte Wert eingegeben werden: Um etwa an der Achse X=100 zu spiegeln, muß für die X-Spiegelachse 200 eingegeben werden.

4. Start Multicolor und **5. Start Hires**:
Drücken Sie die Tasten <4> oder <5>, werden die Bilder berechnet. Sie können dies auf dem Schirm verfolgen. Im Multicolor-Modus ist die Auflösung nicht ganz so gut, dafür werden im Gegensatz zum hochauflösenden Modus (Hires) nicht eine, sondern drei Punktfarben verwendet. Das Bild kann mit der Taste <RUN/STOP> angehalten und mit <Q> ganz abgebrochen werden. Im letzteren Fall erscheint wieder das Menü des Programms.

6. Programmende:
Ein gutes Programm muß natürlich auch wieder verlassen werden können. Nach Betätigung der Taste <6> erscheint ein allgemeinverständlicher Abschiedsgruß, das Programm kann mit RUN wieder gestartet werden.

Leider fehlt dem Programm eine Druckfunktion. Es ist jedoch nicht weiter schwierig, eine Grafik zu drucken. Der Grafikspeicher liegt unter dem Basic-ROM und kann zum Beispiel mit dem »Hardmaker« (64'er-Magazin, Ausgabe 4/87) sehr bequem ausgelesen und gedruckt werden.

Noch ein kleiner Tip für sehr anspruchsvolle Genießer: Wählen Sie eine »weiche« Farbkombination (z.B. Schwarz-Blau) und betrachten Sie die Bildsequenzen in einem abgedunkelten Raum aus einiger Entfernung mit einer guten Tasse Tee. Es lohnt sich!

Hinter den Kulissen

Natürlich ist es sehr interessant, wie so ein Programm arbeitet. Wir haben daher in Listing 3 (nicht abtippen!) die Maschinenroutine aus Listing 2 für Sie dokumentiert. Wenn Sie Ambitionen dazu haben, ähnliche Programme selbst zu programmieren oder sich nur für den Aufbau interessieren, sollten Sie sich Listing 3 genauer ansehen. Es ist recht lehrreich. Genauere Informationen dazu sind nicht mehr notwendig, der Kommentar ist recht genau. Daher soll hier nur ganz grob die Funktionsweise erklärt werden.

Nach dem Aufruf mit SYS 49152 (in Basic, Zeile 970) wird eine Hauptroutine von $c000 bis $c05b aktiviert, die per JSR-Befehl nacheinander verschiedene Hilfsprogramme aufruft.

Zunächst werden die Grafik gelöscht, eingeschaltet und die Farben gesetzt. Bei $c042 wird dann die Haupt-Rechenroutine ab $c19a aufgerufen. Nach dem Abbruch mit <Q> schaltet das Maschinenprogramm die Grafik wieder aus und kehrt zu Basic zurück.

Nun zu den Unterroutinen. Eine zentrale Bedeutung hat die Routine ab $c066, die einen Punkt in der Grafik invertiert. Die Koordinaten werden in den Prozessorregistern übergeben.

Aufgerufen wird die PLOT-Routine von dem Programmteil, der vier symmetrische Punkte setzt. Diese Routine, die das Testen auf den erlaubten Bereich und die Spiegelung an den zwei Achsen durchführt, beginnt bei $c0c4. Von $c132 bis $c199 stehen drei Standardroutinen, die jeder Grafikprogrammierer kennen wird. Sie löschen den Grafikspeicher, der von $a000 bis $bfff unter dem Basic-ROM liegt, schalten diese Grafik ein und setzen die gewählten Farben.

Interessanter ist da schon die Rechenroutine an $c19a. Hier werden zunächst alle vier Timer des C64 program-

Kurzinfo: Computer-Art

Programmart: Grafik-Programm
Laden: LOAD "C-ART",8
Start: Nach dem Laden RUN eingeben. Der Maschinensprache-Teil »C-ART.MC« wird vom Steuerprogramm automatisch nachgeladen.
Besonderheiten: Listing 3 ist der kommentierte Quellcode des Maschinensprache-Programms, den Sie bitte nicht abtippen.
Programmautoren: Claus Faber und Johannes Bernd


```
0 :                                               <232>
10 REM                    A                       <082>
20 REM             COMPUTER                       <238>
30 REM                    T                       <254>
50 REM                                            <112>
60 REM  CLAUS FABER + BERND JOHANNES              <031>
70 REM                                            <132>
80 REM  BEARBEITER: NIKOLAUS HEUSLER              <216>
90 REM                                            <152>
99 REM  FUER 64'ER SONDERHEFT                     <157>
100 REM                                           <162>
101 REM (C) MARKT & TECHNIK                       <136>
102 REM (W) APRIL 1987                            <252>
103 REM     NH-200589-ARR                         <222>
107 :                                             <083>
108 IF PEEK(49152)+PEEK(49153)<>164 THEN L
    OAD"C-ART.MC",8,1                             <193>
109 :                                             <085>
110 BA=49842+PEEK(41234)-61-(480+PEEK(4926
    5)/12-90.75)/SQR(676)                         <048>
130 C1=BA+0:C2=BA+1                               <187>
140 C3=BA+2:C4=BA+3                               <233>
160 R1=BA+4:R2=BA+5                               <089>
170 R3=BA+6:R4=BA+7                               <135>
180 R5=BA+8                                       <246>
200 P1=BA+9:P2=BA+10                              <210>
210 P3=BA+11:P5=BA+12                             <110>
211 P4=BA+13                                      <100>
230 GX=BA+14:GH=BA+15                             <130>
240 GY=BA+16                                      <140>
260 XO=BA+17:XM=BA+18                             <136>
270 HO=BA+19:HM=BA+20                             <184>
280 YO=BA+21:YM=BA+22                             <135>
290 SYS 65409                                     <208>
325 GOSUB 1100                                    <043>
330 POKE 53280,0:POKE 53281,0                     <204>
340 PRINT"{CLR,GREY 3,GRAPHIC,CTRL-H}@@@@@
    @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@";         <169>
342 PRINT"{RVSON,25SPACE,YELLOW}A{GREY 3,1
    4SPACE}";                                     <120>
350 PRINT"{12SPACE}C O M P U T E{SPACE,YEL
    LOW}R{GREY 3,14SPACE}";                       <050>
360 PRINT"{26SPACE,YELLOW}T{GREY 3,12SPACE
    ,WHITE}"                                      <222>
380 PRINT"{2DOWN,SPACE}1) GRAFIK{3SPACE}-P
    ARAMETER                                      <070>
390 PRINT" 2) RECHNUNGS-PARAMETER                 <019>
400 PRINT" 3) PROZESS{2SPACE}-PARAMETER           <146>
410 PRINT"{DOWN,SPACE}4) START (MULTI-COLO
    R)                                            <090>
420 PRINT" 5) START (HOCHAUFLOESEND)              <093>
424 PRINT"{DOWN,SPACE}6) ENDE                     <207>
430 PRINT"{2DOWN,4SPACE}BITTE WAEHLEN             <046>
431 PRINT"{3DOWN,GREY 3}VON C. FABER, B. J
    OHANNES & N. HEUSLER                          <063>
432 PRINT"{DOWN}(C) SH 64'ER, MARKT & TECH
    NIK, 1989{HOME}":POKE 198,.                   <056>
440 GET A$:W=VAL(A$)                              <206>
450 IF W<1 OR W>6 GOTO 440                        <017>
460 ON W GOTO 500,600,700,800,900,1200            <153>
500 PRINT"{CLR,SPACE}GRAFIK-PARAMETER             <008>
510 PRINT"{2DOWN,SPACE}FARBEN:":SR=0:G1=0:
```

miert. Die beiden Timer der CIA 1 dienen hier als Zufallsgeneratoren. Sie werden so eingestellt, daß sie laufend von einem bestimmten, für beide Timer verschiedenen Startwert (über 65000) auf Null herabzählen (diese weite Strecke wird etwa 15mal pro Sekunde durchlaufen). Durch Auslesen dieser Timer an einer bestimmten Stelle im Programm ($c1dd) erhält man zwei Werte, die nicht vorherzusagen sind, also praktisch zufällig. Diese beiden Werte werden in den Bereich 0..255 gebracht und als X- und Y-Koordinaten verwendet.

Die beiden Timer der CIA 2 dienen als echte Timer, sie sind so programmiert, daß sie von einem bestimmten Wert, den Sie selbst als »Bildpause« einstellen, auf Null zählen. Das Programm errechnet erst dann ein neues Bild, wenn die Null erreicht ist. So wird die Pause realisiert.

Ab $c1d7 findet sich eine Schleife, die erst dann beendet wird, wenn die Q-Taste betätigt wird. In dieser Schleife erzeugt der C 64 zunächst die zufälligen Koordinaten, die als Ausgangsbasis für das jetzt zu errechnende Bild dienen. Die Uhr für die Bildpause läuft los. An $c206 wird die Rechenroutine, die bei $c257 liegt, aufgerufen. Diese erzeugt aus den alten Koordinaten neue, die dann in das Bild eingezeichnet die interessanten Figuren ergeben.

Rechne bis zum Umfallen!

Hier werden eigentlich nur die beiden Anfangskoordinaten so oft durch zwei geteilt, wie es die Fract-Parameter festlegen. Die jeweils andere Koordinate wird addiert oder subtrahiert, je nach Vorzeichen. Der Parameter »Krümmung« steuert direkt diese Additionen oder Subtraktionen. Dazu verändert sich das Maschinenprogramm selbst: Am Anfang, ab $c009, werden in dieser Rechenroutine die benötigten Befehle (ADC oder SBC) je nach Benutzervorgabe an bestimmte Platzhalter eingesetzt.

Die Routine prüft nun, ob die Koordinaten, die sie da eben ausgerechnet hat, überhaupt sinnvoll sind. Es kann passieren, daß die errechneten Koordinaten gleich den Ausgangskoordinaten sind. In diesem Fall würde das Programm endlos mit demselben Koordinatenpaar weiterarbeiten, daher wird von der Rechenroutine in diesem Fall das Carry-Flag gesetzt, was dem Hauptprogramm signalisiert, daß die Grafik beendet ist.

Das gleiche passiert, wenn die errechneten Koordinaten mit denen übereinstimmen, die ganz am Anfang ausgelost wurden. Ist »Carry« jedoch gelöscht, sind die neuen Koordinaten zur Weiterführung des Bildes geeignet. Jetzt wird die Routine aufgerufen, die vier Punkte zeichnet. Danach wird wieder die Rechenroutine aufgerufen, und das Spiel geht so weiter, bis der Benutzer abbricht oder dieselben Koordinaten erneut auftreten.

Wenn der Computer erkennt, daß sich die Koordinaten wiederholen, betrachtet er das erzeugte Bild als beendet. Die Routine ab $c234 wartet zunächst, bis das nächste Bild erzeugt werden darf (Bildpause). Soll dieselbe Grafik noch einmal gezeichnet werden, wird die Endlosschleife von vorne gestartet, jedoch ohne Änderung der Startkoordinaten.

Stehen noch »Compound-Grafiken« aus oder soll dieses Bild nicht noch einmal gemalt werden, ruft die Routine den Programmteil wieder auf, der per Zufall neue Koordinaten wählt, und beginnt von vorn.

Die Tasten <RUN/STOP> und <Q> werden direkt über die Tastaturmatrix geprüft (bei $c22d und $c209). Falls sie gedrückt sind, ist das entsprechende Bit (6 oder 7) des Tastenregisters $dc01 gelöscht.

Zu erwähnen sind noch die Variablen, die direkt hinter dem Maschinenprogramm, das logisch bei $c2a7 (auf der Diskette bei $c2a9) endet, angesiedelt sind. Auch sie werden in Listing 3 erklärt. In der Zeropage werden die Zellen $22/23 als Zeiger und $a4 bis $a9 als Koordinatenspeicher (siehe Ende von Listing 3) genutzt.

Aber auch ohne Verständnis der Funktionsweise können Sie sich an der Wirkung, an den Effekten des Programms »laben«. Nehmen Sie sich vielleicht etwas Zeit, und probieren Sie einfach alle Parameter durch. Sie werden sehen: Die Abtipparbeit hat sich gelohnt!

(C. Faber/B. Johannes/N. Heusler/ef)

```
    G2=15                                     <218>
520 PRINT"{2DOWN,SPACE}RAND{7SPACE}";:RE=W
    1:GOSUB 1000:W1=RE                        <016>
530 PRINT"{DOWN,SPACE}HINTERGR.{2SPACE}";:
    RE=W2:GOSUB 1000:W2=RE                    <071>
540 PRINT"{DOWN,SPACE}FARBE 1{4SPACE}";:RE
    =W3:GOSUB 1000:W3=RE                      <057>
550 PRINT"{DOWN,SPACE}FARBE 2{4SPACE}";:RE
    =W4:GOSUB 1000:W4=RE                      <083>
560 PRINT"{DOWN,SPACE}FARBE 3{4SPACE}";:RE
    =W5:GOSUB 1000:W5=RE                      <109>
570 G2=127:PRINT"{DOWN,SPACE}COMPOUND{3SPA
    CE}";:RE=WK:GOSUB 1000:WK=RE              <152>
580 GOTO 330                                  <064>
600 PRINT"{CLR,SPACE}RECHNUNGS-PARAMETER      <240>
610 PRINT"{2DOWN}PARAMETER, DIE IN DIE FUN
    KTION EINGEHEN:{UP}                       <138>
620 SR=0:G1=1:G2=7                            <221>
630 PRINT"{2DOWN,SPACE}X-FRACT. ";:RE=W6:G
    OSUB 1000:W6=RE                           <064>
640 PRINT"{DOWN,SPACE}Y-FRACT. ";:RE=W7:GO
    SUB 1000:W7=RE                            <228>
650 G2=3:PRINT"{DOWN,SPACE}X-VORZEICHEN ";
    :RE=W8:GOSUB 1000:W8=RE                   <166>
660 PRINT"{DOWN,SPACE}Y-VORZEICHEN ";:RE=W
    9:GOSUB 1000:W9=RE                        <129>
670 G1=0:G2=1:PRINT"{DOWN,SPACE}KRUEMMUNG
    ";:RE=WA:GOSUB 1000:WA=RE                 <241>
680 GOTO 330                                  <164>
700 PRINT"{CLR,SPACE}PROZESS-PARAMETER        <005>
710 PRINT"{2DOWN,SPACE}ABLAUFSTEUERNDE PAR
    AMETER:                                   <205>
720 G1=1:G2=255:SR=0                          <173>
730 PRINT"{2DOWN,SPACE}BILDWIEDERHOLUNG ";
    :RE=WB:GOSUB 1000:WB=RE                   <252>
740 PRINT"{DOWN,SPACE}FARBSTETIGKEIT{3SPAC
    E}";:RE=WC:GOSUB 1000:WC=RE               <162>
750 PRINT"{DOWN,SPACE}BILDPAUSE{8SPACE}";:
    RE=WD:GOSUB 1000:WD=RE                    <077>
760 G2=320:PRINT"{DOWN,SPACE}X-GRENZE ";:R
    E=WE+WF*256:GOSUB 1000:WF=INT(RE/256)     <039>
770 WE=RE AND 255:G2=200:PRINT"{DOWN,SPACE
    }Y-GRENZE ";:RE=WG:GOSUB 1000:WG=RE:G2
    =320                                      <146>
780 PRINT"{DOWN,SPACE}X-SPIEGELACHSE ";:RE
    =WH+WI*256:GOSUB 1000:WH=RE AND 255:WI
    =INT(RE/256)                              <103>
790 G2=200:PRINT"{DOWN,SPACE}Y-SPIEGELACHS
    E ";:RE=WJ:GOSUB 1000:WJ=RE:GOTO 330      <123>
800 POKE P4,0:POKE C1,W1:POKE C2,W2:POKE C
    3,W3:POKE C4,W4*16+W5                     <054>
810 TT=2:GOTO 920                             <067>
900 POKE P4,128:POKE C1,W1:POKE C4,W3*16+W
    2                                         <201>
910 TT=1                                      <216>
920 POKE R1,W6:POKE R2,W7:POKE R3,W8:POKE
    R4,W9:POKE R5,WA                          <044>
930 POKE P1,WC:POKE P2,WB:POKE P3,WD          <139>
940 POKE GX,((WE+WF*256)/TT)AND 255:POKE G
```

Listing 1. Das Steuerprogramm ist in Basic verfaßt

```
    H,WF/TT:POKE GY,WG                          <055>
950 POKE XO,WH:POKE XM,((WH+WI*256)/2)AND
    255:POKE HO,WI:POKE HM,WI/2                 <116>
960 POKE YO,WJ:POKE YM,WJ:POKE P5,WK            <062>
970 SYS 49152                                   <010>
999 GOTO 330                                    <229>
1000 IF SR<0 THEN PRINT"("G1$"-"G2$"):";:G
     OTO 1009                                   <011>
1003 G1$=STR$(G1):G2$=STR$(G2)                  <002>
1006 PRINT"("RIGHT$(G1$,LEN(G1$)-1)"-"RIGH
     T$(G2$,LEN(G2$)-1)"):";                    <041>
1009 A=PEEK(211)                                <105>
1010 IF SR<0 THEN PRINT RE$;:GOTO 1030          <247>
1020 PRINT RE;                                  <181>
1030 POKE 211,A-1                               <238>
1040 INPUT RE$:RE=VAL(RE$)                      <187>
1050 IF SR<0 THEN RETURN                        <242>
1060 IF RE<G1 OR RE>G2 THEN PRINT"{2UP}":G
     OTO 1030                                   <235>
1070 RETURN                                     <112>
1100 W1=0:W2=0:W3=1:W4=5:W5=2                   <087>
1110 W6=2:W7=2:W8=1:W9=1:WA=1                   <079>
1120 WB=3:WC=1:WD=18:WE=160:WF=0:WG=100:WH
     =64:WI=1:WJ=200:WK=0                       <120>
1130 REM DEFAULTWERTE KOENNEN NACH BELIEBE
     N GEAENDERT WERDEN !                       <175>
1199 RETURN                                     <241>
1200 PRINT"{CLR}TSCHUESS / SERVUS / SALUT
     / ADE / CIAOU":END                         <173>
```

Listing 1. (Schluß)

```
Name : c-art.mc              c000 c2aa
----------------------------------
c000 : 20 84 ff ad b8 c2 29 03  64
c008 : aa bd 5c c0 8d 66 c2 ad  32
c010 : b9 c2 29 03 aa bd 5c c0  61
c018 : 8d 72 c2 ad ba c2 29 01  ad
c020 : aa 49 01 a8 bd 64 c0 8d  e1
c028 : 67 c2 b9 64 c0 8d 73 c2  b7
c030 : a9 36 85 01 a9 00 8d ae  a4
c038 : c2 20 32 c1 20 73 c1 20  b4
c040 : 4a c1 20 9a c1 20 84 ff  f5
c048 : a9 1b 8d 11 d0 a9 c8 8d  9d
c050 : 16 d0 a9 15 8d 18 d0 a9  0c
c058 : 37 85 01 60 ea ea 38 18  b5
c060 : 04 04 e9 69 e5 65 2c bf  c8
c068 : c2 30 06 48 8a 0a aa 68  41
c070 : 2a 09 a0 85 23 98 29 07  a1
c078 : 85 22 8a 29 f8 05 22 85  21
c080 : 22 98 4a 4a 4a 85 24 4a  c0
c088 : 4a 18 65 23 65 24 85 23  70
c090 : 98 29 f8 0a 0a 0a 18 65  58
c098 : 22 85 22 90 02 e6 23 a5  47
c0a0 : 23 c9 a0 90 1e c9 c0 b0  77
c0a8 : 1a 8a 29 07 2c bf c2 30  5f
c0b0 : 08 4a ae ae c2 18 7d 91  65
c0b8 : c2 aa bd 94 c2 a0 00 51  a5
c0c0 : 22 91 22 60 8e a8 c2 8d  94
c0c8 : aa c2 8c ac c2 ec c0 c2  a8
c0d0 : ed c1 c2 b0 5c cc c2 c2  21
c0d8 : b0 57 ad bf c2 49 80 0a  24
c0e0 : a9 00 2a aa bd c7 c2 38  ff
c0e8 : ed ac c2 8d ad c2 bd c3  fd
c0f0 : c2 38 ed a8 c2 8d a9 c2  23
c0f8 : bd c5 c2 ed aa c2 8d ab  54
c100 : c2 ad aa c2 ae a8 c2 ac  30
c108 : ac c2 20 66 c0 ad aa c2  94
c110 : ae a8 c2 ac ad c2 20 66  97
c118 : c0 ad ab c2 ae a9 c2 ac  8f
c120 : ac c2 20 66 c0 ad ab c2  b0
c128 : ae a9 c2 ac ad c2 4c 66  e0
c130 : c0 60 a9 00 a0 a0 85 22  f4
c138 : 84 23 a2 20 a0 00 98 91  8a
c140 : 22 c8 d0 fb e6 23 ca d0  ce
c148 : f6 60 a9 01 8d 00 dd a9  9c
c150 : 38 8d 18 d0 ad b2 c2 8d  06
c158 : 20 d0 ad b3 c2 8d 21 d0  81
c160 : a9 3b 8d 11 d0 2c bf c2  1f
c168 : 30 08 ad 16 d0 09 10 8d  7b
c170 : 16 d0 60 a0 00 ad b4 c2  e0
c178 : 99 00 d8 99 00 d9 99 00  b0
c180 : da 99 00 db c8 d0 f1 ad  d8
c188 : b5 c2 99 00 8c 99 00 8d  b5
c190 : 99 00 8e 99 00 8f c8 d0  41
c198 : f1 60 a9 01 8d 0e dc 8d  1c
c1a0 : 0f dc a9 ff 8d 04 dc 8d  0f
c1a8 : 05 dc 8d 07 dc a9 fe 8d  92
c1b0 : 06 dc a9 08 8d 0e dd a9  a4
c1b8 : 48 8d 0f dd ad bd c2 8d  35
c1c0 : 06 dd a9 00 8d 07 dd a9  fb
c1c8 : 55 8d 04 dd a9 20 8d 05  7c
c1d0 : dd ad be c2 8d b0 c2 ad  50
c1d8 : bc c2 8d af c2 ad 04 dc  b2
c1e0 : 0d 07 dc 85 a4 85 a5 ad  c1
c1e8 : 06 dc 0d 05 dc 85 a7 85  e4
c1f0 : a8 a9 49 8d 0f dd a9 01  f9
c1f8 : 8d 0e dd a9 00 8d ae c2  e6
c200 : ad bb c2 8d b1 c2 20 57  4d
c208 : c2 2c 01 dc 50 48 b0 24  0f
c210 : 20 c4 c0 ce b1 c2 d0 15  3b
c218 : ad bb c2 8d b1 c2 ad ae  4a
c220 : c2 18 69 01 c9 03 90 02  64
c228 : a9 00 8d ae c2 2c 01 dc  56
c230 : 10 fb 30 d2 ad 0f dd 29  c1
c238 : 01 d0 f9 a5 a5 85 a4 a5  39
c240 : a8 85 a7 ad b0 c2 f0 06  3b
c248 : ce b0 c2 4c d7 c1 ce af  cf
c250 : c2 d0 9e 4c d7 c1 60 a5  04
c258 : a4 85 a6 a6 a7 86 a9 ac  ec
c260 : b6 c2 4a 88 d0 fc ea 65  86
c268 : a7 85 a4 ac b7 c2 4a 88  5c
c270 : d0 fc ea e5 a6 85 a7 a9  be
c278 : 00 a6 a4 a4 a7 e4 a5 d0  63
c280 : 06 c4 a8 d0 02 38 60 e4  5a
c288 : a6 d0 04 c4 a9 f0 f6 18  5e
c290 : 60 08 0c 10 80 40 20 10  a4
c298 : 08 04 02 01 40 10 04 01  da
c2a0 : 80 20 08 02 c0 30 0c 03  36
c2a8 : 00 00 ff ff ff ff 00 00  a8
```

Listing 2. Geben Sie das Maschinenprogramm mit dem MSE (Seite 159) ein

```
                 ; Programm von Claus Faber &
                   Bernd Johannes
                 ; Listing-Kommentar von
                   Nikolaus Heusler
                 ; (c)opyright Markt&Technik, SH 64'er
                 ; von hier Einsprung durch SYS 49152

$c000 jsr $ff84  ; CIAs und Interrupts initialisieren
$c003 lda $c2b8  ; X-Vorzeichen
$c006 and  #$03  ; erlaubter Bereich: 1 bis 3
                   (eigentl. unnötig)
$c008 tax        ; nach x als Zähler
$c009 lda $c05c,x ; aus Tabelle zugehörigen Befehl lesen
$c00c sta $c266  ; in Programm schreiben
$c00f lda $c2b9  ; Y-Vorzeichen
$c012 and  #$03  ; erlaubter Bereich: 1 bis 3
                   (eigentl. unnötig)
$c014 tax        ; nach x als Zähler
$c015 lda $c05c,x ; aus Tabelle zugehörigen Befehl lesen
$c018 sta $c272  ; in Programm schreiben
$c01b lda $c2ba  ; Krümmung
$c01e and  #$01  ; erlaubter Bereich: 0 und 1
$c020 tax        ; nach x
$c021 eor  #$01  ; 0 -> 1 und 1 -> 0
$c023 tay        ; invertierten Wert nach y
$c024 lda $c064,x ; ADC oder SBC-Befehl holen
$c027 sta $c267  ; in Programm schreiben
$c02a lda $c064,y ; anderen Befehl holen
$c02d sta $c273  ; und auch fixieren
$c030 lda  #$36  ; RAM-Bereich bei $a000-$bfff
                   einschalten
$c032 sta   $01  ; in Prozessorport (Basic abschalten)
$c034 lda  #$00  ; Farbzähler
$c036 sta $c2ae  ; auf Null stellen
$c039 jsr $c132  ; Grafikspeicher löschen
$c03c jsr $c173  ; Farben setzen (Farbrams)
$c03f jsr $c14a  ; Grafik einschalten
$c042 jsr $c19a  ; Grafik berechnen
```

```
$c045 jsr $ff84    ; CIAs initialisieren (wurden ja
                     für Grafik >>mißbraucht<<)
$c048 lda  #$1b    ; dezimal 27
$c04a sta $d011    ; Grafik abschalten
$c04d lda  #$c8    ; dezimal 200
$c04f sta $d016    ; Multicolor ausschalten
$c052 lda  #$15    ; dezimal 21
$c054 sta $d018    ; Textmodus einschalten
$c057 lda  #$37    ; Basic einschalten
$c059 sta  $01     ; Prozessorport
$c05b rts          ; zurück zum Basic-Hauptprogramm
                   ; drei Befehle für drei Vorzeichen:
                   ; 0/1 = NOP
                   ;   2 = SEC
                   ;   3 = CLC
$c05c b  $ea $ea $38 $18
                   ; $c05f bis $c063 unbenutzt
$c05f b  $04 $04 $e9 $69
$c064 b  $e5 $65   ; zwei Befehle für die Krümmung
                   ;   0 = SBC
                   ;   1 = ADC
                   ; Punkt in Grafik invertieren (PLOT)
                   ; Parameter: X-Koordinate:
                     X-Register (low), Akku (high)
                     Y-Koordinate: Y-Register
$c066 bit $c2bf    ; Multicolor-Modus ?
$c069 bmi $c071    ; nein
                   ; sonst X-Koordinate mit 2
                     multiplizieren
$c06b pha          ; Highbyte merken
$c06c txa          ; Lowbyte nach A
$c06d asl          ; mal 2
$c06e tax          ; zurück nach X
$c06f pla          ; Highbyte
$c070 rol          ; mal 2 (ggf. Übertrag von Lowbyte)
                   ; Adresse = $a000 + (X and 256) + 40
                     * Y + (Y and 7) + 8 * int (X/8) +
                     64 * int (Y/8)
                   ; grahi = $a000 + (X and 256)
$c071 ora  #$a0    ; plus Startadresse des Grafikspeichers
$c073 sta  $23     ; merken; grahi = $a000 + (X and 256)
                   ; offset = (Y and 7) + 8 * int (X/8)
$c075 tya          ; Y-Koordinate
$c076 and  #$07    ; untere 3 Bits
$c078 sta  $22     ; merken
$c07a txa          ; X-Koordinate
$c07b and  #$f8    ; obere 5 Bits: a = 8 * int (X/8)
$c07d ora  $22     ; plus (Y-Koordinate and 7)
$c07f sta  $22     ; als offset merken
                   ; graf = grahi + 256 * (5/32) * Y
                   ; oder graf = grahi + 40 * Y
$c081 tya          ; Y-Koordinate
$c082 lsr
$c083 lsr          ; durch 8 teilen
$c084 lsr
$c085 sta  $24     ; merken
$c087 lsr          ; und weiter durch 4 teilen
$c088 lsr          ; 1/8 + 1/32 = 5/32
$c089 clc          ; Addition vorbereiten
$c08a adc  $23     ; plus Highbyte der Adresse
$c08c adc  $24     ; plus Y-Koordinate durch 8
$c08e sta  $23     ; gibt neues Adress-Highbyte (graf)
                   ; Adresse = graf + offset + 64 * int
                     (Y/8)
$c090 tya          ; Y-Koordinate
$c091 and  #$f8    ; obere 5 Bits isolieren: a = 8 * int
                     (Y/8)
$c093 asl
$c094 asl          ; mal 8
$c095 asl          ; ergibt 64 * int (Y/8)
$c096 clc          ; Addition vorbereiten
$c097 adc  $22     ; plus offset
$c099 sta  $22     ; gibt Adresse
$c09b bcc $c09f    ; kein Übertrag ?
$c09d inc  $23     ; sonst auch Highbyte erhöhen
                   ; testen, ob Adresse zwischen $a000
                     und $bfff liegt
$c09f lda  $23     ; Highbyte
$c0a1 cmp  #$a0    ; kleiner $a000 ?
$c0a3 bcc $c0c3    ; ja, Routine abbrechen
$c0a5 cmp  #$c0    ; groesser gleich $c000 ?
$c0a7 bcs $c0c3    ; ja, Routine abbrechen
                   ; Adresse der Grafikspeicherzelle
                     in $22/23
$c0a9 txa          ; X-Koordinate
$c0aa and  #$07    ; >>Feineinstellung<<, Nummer
                     des Bits im Byte
$c0ac bit $c2bf    ; Multicolor-Modus ?
$c0af bmi $c0b9    ; nein
$c0b1 lsr          ; sonst durch 2 teilen
$c0b2 ldx $c2ae    ; Farbnummer
$c0b5 clc
$c0b6 adc $c291,x  ; entspr. Bitkombination addieren
$c0b9 tax          ; X = Nummer des Bits im Byte
$c0ba lda $c294,x  ; A = 2 hoch X (HiRes)
$c0bd ldy  #$00    ; Hilfszeiger
$c0bf eor ($22),y  ; Pixel in der Grafik invertieren
$c0c1 sta ($22),y  ; und wieder in Grafik schreiben
$c0c3 rts          ; fertig
                   ; vier Punkte (Punkte-Quartett) setzen
                   ; Parameter: X-Koordinate:
                     X-Register (low), Akku (high)
                     Y-Koordinate: Y-Register
$c0c4 stx $c2a8    ; X-Koordinate low merken
$c0c7 sta $c2aa    ; X-Koordinate high
                     merken (ist immer Null, siehe $c277)
$c0ca sty $c2ac    ; Y-Koordinate merken
                   ; liegt die X-Koordinate in
                     den erlaubten Grenzen ?
$c0cd cpx $c2c0    ; Low-Byte
$c0d0 sbc $c2c1    ; mit Sollwert vergleichen
$c0d3 bcs $c131    ; zu gross, dann Routine abbrechen
                   ; liegt die Y-Koordinate im
                     erlaubten Bereich ?
$c0d5 cpy $c2c2    ; mit Sollwert vergleichen
$c0d8 bcs $c131    ; zu gross, dann Abbruch
                   ; gespiegelte Koordinaten berechnen
$c0da lda $c2bf    ; Multicolor-Modus ?
$c0dd eor  #$80    ; Multicolorbit umdrehen
$c0df asl          ; und ins Carry
$c0e0 lda  #$00    ; unnötig: A wird bei $c0df Null
$c0e2 rol          ; A = 1 => Multicolor, sonst Null
                     (HiRes)
$c0e3 tax          ; als Zähler merken
$c0e4 lda $c2c7,x  ; Y-Spiegelachse lesen ($c2c7 HiRes,
                     $c2c8 Multi.)
$c0e7 sec          ; für Subtraktion
$c0e8 sbc $c2ac    ; davon die Y-Koordinate abziehen
$c0eb sta $c2ad    ; gibt neue Y-Koordinate
$c0ee lda $c2c3,x  ; X-Spiegelachse analog zu $c0e4
                     lesen (low)
$c0f1 sec          ; Subtraktion vorbereiten
$c0f2 sbc $c2a8    ; davon X-Koordinate abziehen
$c0f5 sta $c2a9    ; gibt neue X-Koordinate (low)
$c0f8 lda $c2c5,x  ; X-Spiegelachse high
$c0fb sbc $c2aa    ; davon Highbyte der X-Koordinate
                     abziehen
$c0fe sta $c2ab    ; gibt neue X-Koordinate (high)
                   ; vier Punkte setzen:
                   ; X-Original, Y-Original
$c101 lda $c2aa
$c104 ldx $c2a8
$c107 ldy $c2ac
$c10a jsr $c066    ; PLOT, Punkt in Grafik invertieren
                   ; X-Original, Y-gespiegelt
$c10d lda $c2aa
$c110 ldx $c2a8
$c113 ldy $c2ad
$c116 jsr $c066    ; PLOT, Punkt in Grafik invertieren
                   ; X-gespiegelt, Y-Original
```

Listing 3.
Hier sehen Sie Listing 2 als Klartext, genau kommentiert



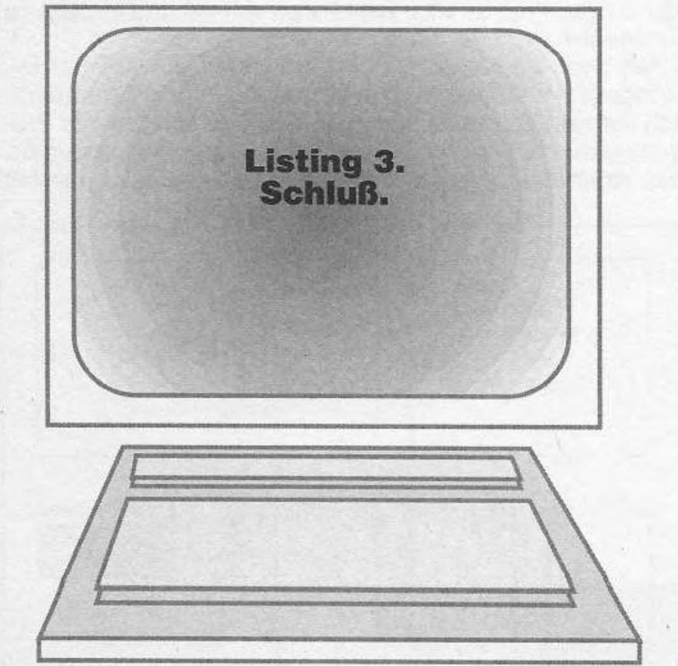

64ER ONLINE

```
$c119 lda $c2ab
$c11c ldx $c2a9
$c11f ldy $c2ac
$c122 jsr $c066       ; PLOT, Punkt in Grafik invertieren
                      ; X-gespiegelt, Y-gespiegelt
$c125 lda $c2ab
$c128 ldx $c2a9
$c12b ldy $c2ad
$c12e jmp $c066       ; PLOT, Punkt in Grafik invertieren
$c131 rts
                      ; Grafikspeicher $a000-$bfff löschen
$c132 lda  #00        ; Lowbyte Grafikspeicheradresse
$c134 ldy  #$a0       ; und Highbyte ($a000)
$c136 sta  $22        ; nach $22/23
$c138 sty  $23
$c13a ldx  #$20       ; 32 pages löschen
$c13c ldy  #$00       ; tay ginge auch
$c13e tya             ; unnötig ; A ist schon 0
$c13f sta ($22),y     ; Null in Grafikspeicher
$c141 iny             ; nächstes Byte
$c142 bne $c13f       ; noch kein Übertrag
$c144 inc  $23        ; Highbyte erhöhen
$c146 dex             ; noch eine Seite ?
$c147 bne $c13f       ; ja
$c149 rts             ; sonst Grafik komplett gelöscht
                      ; Grafik einschalten
$c14a lda  #$01       ; Bank $8000 einschalten
$c14c sta $dd00       ; Video-Bank-Register
$c14f lda  #$38       ; dezimal 56
$c151 sta $d018       ; HiRes-Schirm bei $a000
$c154 lda $c2b2       ; Randfarbe
$c157 sta $d020       ; in VIC schreiben
$c15a lda $c2b3       ; Hintergrundfarbe
$c15d sta $d021       ; in VIC schreiben
$c160 lda  #$3b       ; dezimal 59
$c162 sta $d011       ; Grafikmodus einschalten
$c165 bit $c2bf       ; Multicolor-Modus ?
$c168 bmi $c172       ; nein
$c16a lda $d016
$c16d ora  #$10       ; dezimal 16
$c16f sta $d016       ; Multicolor-Modus einschalten
$c172 rts             ; und zurück
                      ; Farbrams füllen
$c173 ldy  #$00       ; Zählregister
$c175 lda $c2b4       ; Farbenregister
$c178 sta $d800,y     ; ins Farbram
$c17b sta $d900,y     ; von $d800-$dbff
$c17e sta $da00,y     ; schreiben
$c181 sta $db00,y     ; und somit Farbe setzen
$c184 iny             ; nächstes Byte
$c185 bne $c178       ; noch nicht fertig
$c187 lda $c2b5       ; Hintergrund/Vordergrundfarbe (y=0)
$c18a sta $8c00,y
$c18d sta $8d00,y     ; in Textbildschirm von Page $8000
$c190 sta $8e00,y     ; schreiben
$c193 sta $8f00,y     ; und somit auch diese Farben setzen
$c196 iny             ; nächstes Byte
$c197 bne $c18a       ; noch nicht fertig
$c199 rts             ; sonst zurück
                      ; Hauptroutine: GRAFIK BERECHNEN
                      ; Timer A & B von CIA1 dienen als Zu-
                      fallsgeneratoren für X/Y-Koordinaten
$c19a lda  #$01       ; Zufallsgeneratoren starten (CIA 1),
                      sollen Systemtakte zählen
$c19c sta $dc0e       ; Generator 1 und 2 (Timer A)
                      free running
$c19f sta $dc0f       ; Generator 3 und 4 (Timer B)
                      free running
$c1a2 lda  #$ff       ; Startwerte für Zufallsgeneratoren
$c1a4 sta $dc04       ; Timer A low
$c1a7 sta $dc05       ; Timer A high
$c1aa sta $dc07       ; Timer B high
$c1ad lda  #$fe       ; damit die beiden Generatoren
                      asynchron laufen
$c1af sta $dc06       ; Timer B low
                      ; Timer A & B von CIA2 dienen für
                      Bildpause
$c1b2 lda  #$08       ; Timer A (CIA 2), soll Systemtakte
                      zählen
$c1b4 sta $dd0e       ; in CIA 2
$c1b7 lda  #$48       ; Timer B (CIA 2) one shot, soll
                      Underflows von Timer A zählen
$c1b9 sta $dd0f       ; in CIA 2
$c1bc lda $c2bd       ; Bildpause
$c1bf sta $dd06       ; als Startwert Timer B (low)
$c1c2 lda  #$00       ; Startwert Timer B high
$c1c4 sta $dd07       ; ist immer Null
$c1c7 lda  #$55       ; dezimal 85
$c1c9 sta $dd04       ; Timer A high (CIA 2)
$c1cc lda  #$20       ; dezimal 32 (gibt dezimal 8277)
$c1ce sta $dd05       ; Timer A low (CIA 2)
                      ; gibt Zeitbasis ca. 1/120 Sekunde
$c1d1 lda $c2be       ; Compound
$c1d4 sta $c2b0       ; merken
                      ; Beginn der Endlosschleife:
                      neues Bild erzeugen
$c1d7 lda $c2bc       ; Bildwiederholung
$c1da sta $c2af       ; merken
                      ; zufällige X/Y-Koordinaten erzeugen
$c1dd lda $dc04       ; Zufallswert 1 lesen (Timer A low)
$c1e0 ora $dc07       ; mit Zufallswert 2 verknüpfen
                      (Timer B high)
$c1e3 sta  $a4        ; (nx) gibt X-Koordinate (neu)
$c1e5 sta  $a5        ; (ox) und X-Koordinate (Anfang)
$c1e7 lda $dc06       ; Zufallswert 3 lesen (Timer B low)
$c1ea ora $dc05       ; mit Zufallswert 4 verknüpfen
                      (Timer A high)
$c1ed sta  $a7        ; (ny) gibt Y-Koordinate (neu)
$c1ef sta  $a8        ; (oy) und Y-Koordinate (Anfang)
                      ; selbes Bild nochmal
$c1f1 lda  #$49       ; Bit 1 setzen: Uhr starten
$c1f3 sta $dd0f       ; Timer B (CIA 2) starten
$c1f6 lda  #$01       ; Timer A (CIA 2) starten
$c1f8 sta $dd0e       ; Uhr läuft
$c1fb lda  #$00
$c1fd sta $c2ae       ; Farbenzähler auf Null
$c200 lda $c2bb       ; Farbstetigkeit
$c203 sta $c2b1       ; merken
                      ; Grafik weiterführen
$c206 jsr $c257       ; Rechenroutine; Koordinaten
                      nach X/A und Y
$c209 bit $dc01       ; Taste <Q> prüfen
$c20c bvc $c256       ; gedrückt, dann Endlosschleife
                      beenden (RTS)
$c20e bcs $c234       ; Carry Flag ? dann hat Rechenroutine
                      >>schlecht<< gearbeitet
                      (siehe $c285 bzw. $c28f)
                      ; CARRY GELÖSCHT => Rechenroutine hat
                      >>ordentlich<< gearbeitet →
                      Bild darf weitergeführt werden
$c210 jsr $c0c4       ; Punkte-Quartett setzen
$c213 dec $c2b1       ; Farbnummernzähler herunterzählen
$c216 bne $c22d       ; nicht null, nochmal die selbe Farbe
                      ; neue Farbe
$c218 lda $c2bb       ; Farbstetigkeit
$c21b sta $c2b1       ; als Farbnummernzähler merken
$c21e lda $c2ae       ; Farbenzähler
$c221 clc             ; für Addition
$c222 adc  #$01       ; plus 1
$c224 cmp  #$03       ; schon alle vier Farben ?
$c226 bcc $c22a       ; nein, nächste Farbe
$c228 lda  #$00       ; sonst wieder 1. Farbe
$c22a sta $c2ae       ; wählen
                      ; Farb-Frage geklärt
$c22d bit $dc01       ; <RUN/STOP> Taste testen
$c230 bpl $c22d       ; solange gedrückt, weiter warten
$c232 bmi $c206       ; unbedingter Sprung
                      ; CARRY GESETZT => >>Fehler<< aus
                      Rechenroutine => Bild beendet
$c234 lda $dd0f       ; CIA 2, Timer B
$c237 and  #$01       ; Timer läuft noch ?
$c239 bne $c234       ; ja, Bildpause dauert noch an, also
                      warten bis Ende der Pause
$c23b lda  $a5        ; (ox) alte X-Koordinate
```

```
$c23d sta   $a4   ; (nx) wird zur neuen X-Koordinate
$c23f lda   $a8   ; (oy) alte Y-Koordinate
$c241 sta   $a7   ; (ny) wird zur neuen Y-Koordinate
$c243 lda $c2b0   ; Compound-Zähler
$c246 beq $c24e   ; keine Grafik mehr fixieren?
$c248 dec $c2b0   ; nächste Grafik fixieren
$c24b jmp $c1d7   ; weiter in Endlosschleife
                  ; keine Grafik soll mehr fixiert werden
$c24e dec $c2af   ; nächstes (anderes) Bild erzeugen ?
$c251 bne $c1f1   ; nein, selbes nochmal
$c253 jmp $c1d7   ; sonst weiter in Endlosschleife
$c256 rts         ; Ende
                  ; Rechenroutine: Koordinaten des
                    nächsten Bildes aus alten Koordinaten
                    berechnen
                  ; nx = nx / (2 hoch X-Fract) +/- ny
$c257 lda   $a4   ; nx  ; Koordinaten beim Eintritt in
                    Rechenroutine merken
$c259 sta   $a6   ; vx
$c25b ldx   $a7   ; ny
$c25d stx   $a9   ; vy
$c25f ldy $c2b6   ; X-Fract = Anzahl der Teilungen
$c262 lsr         ; geteilt durch 2
$c263 dey
$c264 bne $c262   ; weiter teilen
                  ; an dieser Stelle wird, je nach
                    X-Vorzeichen und Krümmung, der
                    passende Befehl eingebaut
$c266 nop         ; Selbstmodifikation bei $c00c
$c267 adc   $a7   ; ny, Selbstmodifikation bei $c027
$c269 sta   $a4   ; nx, neue X-Koordinate
                  ; ny = nx / (2 hoch Y-Fract) +/- vx
$c26b ldy $c2b7   ; Y-Fract = Anz. der Teilungen
$c26e lsr         ; geteilt durch 2
$c26f dey
$c270 bne $c26e   ; weiter teilen
                  ; an dieser Stelle wird, je nach
                    Y-Vorzeichen und Krümmung, der
                    passende Befehl eingebaut
$c272 nop         ; Selbstmodifikation bei $c018
$c273 sbc   $a6   ; vx, Selbstmodifikation bei $c02d
$c275 sta   $a7   ; ny, neue Y-Koordinate
$c277 lda  #$00   ; X-Koordinate high: Null
$c279 ldx   $a4   ; nx, neue X-Koordinate
$c27b ldy   $a7   ; ny, neue Y-Koordinate
$c27d cpx   $a5   ; (ox) neue X-Koordinate gleich alter
                    X-Koordinate ?
$c27f bne $c287   ; nein
$c281 cpy   $a8   ; (oy) neue Y-Koordinate gleich alter
                    Y-Koordinate ?
$c283 bne $c287   ; nein
$c285 sec         ; die gleichen Koordinaten wie zu
                    Beginn der Figur
                    => Fehler
$c286 rts         ; und fertig; X-Koordinate low in X,
                    Y-Koordinate in Y
$c287 cpx   $a6   ; (vx) neue X-Koordinate gleich
                    X-Koordinate bei Eintritt in
                    Rechenroutine?
$c289 bne $c28f   ; nein, OK
$c28b cpy   $a9   ; (vy) neue Y-Koordinate gleich
                    Y-Koordinate bei Eintritt in
                    Rechenroutine?
$c28d beq $c285   ; ja, Fehler
$c28f clc         ; Carry wird genau dann gelöscht, wenn
                    neues Bild weder die gleichen
                    Koordinaten
$c290 rts         ; wie letztes Bild noch wie allererstes
                    Bild hat (dann dürfen die neuen
                    Koordinaten verwendet werden)
                  ; Ende des Maschinenprogramms
                  ; Additionswerte für drei Farben
$c291 b  $08 $0c $10
                  ; Tabelle mit Zweierpotenzen (HiRes)
$c294 b  $80 $40 $20 $10 $08 $04 $02 $01
                  ; Tabelle mit Zweierpotenzen (Multi)
$c29c b  $40 $10 $04 $01 $80 $20 $08 $02 $c0 $30 $0c $03
                  ; Variablen
```

$c29c b $40 $10 $04 $01 $80 $20 $08 $02 $c0 $30 $0c $03
; Variablen

```
$c2a8 X-Koordinate low Original
$c2a9 X-Koordinate low gespiegelt
$c2aa X-Koordinate high Original
$c2ab X-Koordinate high gespiegelt
$c2ac Y-Koordinate Original
$c2ad Y-Koordinate gespiegelt
$c2ae Farbzähler
$c2af Bildwiederholungszähler
$c2b0 Compoundzähler
$c2b1 Zähler: Wie oft noch diese Farbe?
$c2b2 Rahmenfarbe
$c2b3 Hintergrundfarbe
$c2b4 Farben 2 & 3
$c2b5 Vordergrund/Hintergrundfarbe
$c2b6 X-Fract
$c2b7 Y-Fract
$c2b8 X-Vorzeichen
$c2b9 Y-Vorzeichen
$c2ba Krümmung
$c2bb Farbstetigkeit
$c2bc Bildwiederholung: Wie oft ein Bild wiederholen ?
$c2bd Bildpause
$c2be Compound: Anz. der zu fixierenden Bilder
$c2bf Modus: 128 = Hires, 0 = Multicolor
$c2c0 X-Grenze low
$c2c1 X-Grenze high
$c2c2 Y-Grenze
$c2c3 X-Spiegelachse low Hires
$c2c4 X-Spiegelachse low Multi
$c2c5 X-Spiegelachse high Hires
$c2c6 X-Spiegelachse high Multi
$c2c7 Y-Spiegelachse Hires
$c2c8 Y-Spiegelachse Multi
                  ; in der Zeropage werden diese Zellen
                    als Koordinatenspeicher benutzt:
$a4: nx, enthält neue X-Koordinate
$a5: ox, enthält X-Koordinate, mit der die Figur
         begonnen wurde
$a6: vx, enthält X-Koordinate des unmittelbar
         letzten Punktes
$a7: ny, enthält neue Y-Koordinate
$a8: oy, enthält Y-Koordinate, mit der die Figur
         begonnen wurde
$a9: vy, enthält Y-Koordinate des unmittelbar letzten
         Punktes
```



Universeller Editor

Zeichensätze entwerfen – das klingt eigentlich ganz simpel. Daß dadurch auch das Programmieren schneller Spiele zum Kinderspiel wird, beweist dieses Programm. Aber lesen Sie selbst, welche enormen Leistungen geboten werden.

Mit »Character-Editor« (Listing 1) können sowohl ein- als auch mehrfarbige Zeichensätze entworfen werden. Wie wir Ihnen gleich zeigen werden, können Sie auf diese Weise ausgezeichnet sogenannte »Playfields« entwickeln, also Hintergrundgrafiken für Spiele. Das Programm ist vollständig in Maschinensprache geschrieben und daher extrem schnell. Jede nur erdenkliche Manipulation eines Zeichens ist damit problemlos möglich. Geladen wird der Editor mit

```
LOAD "CHARACTER-EDITOR",8,1
```

und mit RUN gestartet. Nach einem <RUN/STOP RESTORE> kann der »Character-Editor« mit RUN erneut gestartet werden, nach einem Reset mit SYS 2083. Der alte Zeichensatz bleibt beim Neustart erhalten, Feld 2 wird allerdings gelöscht. Bei Sicherheitsabfragen (beispielsweise »löschen?« oder »wirklich?«) bedeutet <J> »Ja« und jede andere Taste »Nein«. Menüs können mit <RUN/STOP>, <←> oder <SPACE> verlassen werden.

Die einzelnen Menüpunkte erscheinen beim Aufruf als Window im unteren Bereich des Bildschirms. Dort werden ansonsten drei Zeichen angezeigt, und zwar das Zeichen aus Feld 1 (mittleres Zeichen), aus Feld 2 (rechtes Zeichen) und das zu bearbeitende Zeichen (linkes Zeichen). Daneben finden Sie natürlich auch eine Anzeige der gewählten Farben und Modi.

In Bild 1 sehen Sie die Bildschirmaufteilung des Editors, in Tabelle 1 eine Erklärung der einzelnen Felder. Eine detaillierte Beschreibung der Befehle für jedes Feld entnehmen Sie bitte Tabelle 2. Dort finden Sie auch alle Punkte zusammengefaßt, die besonders beachtet werden müssen.

Da Bilder bekanntlich mehr sagen als tausend Worte, haben wir für Sie einige wichtige Funktionen direkt vom Bildschirm abfotografiert. Nach Erscheinen der Einschaltmeldung (bitte drücken Sie die SPACE-Taste) sehen Sie den Bildschirmaufbau (Bild 2). Jetzt können Sie das Lade-Menü aufrufen und einen Zeichensatz laden, falls sich auf der Diskette bereits ein fertiger Zeichensatz befindet (Bild 3). Im Disk-Menü (Bild 4) kann beispielsweise der Fehlerkanal der Floppy abgefragt werden (Bild 5). Umfangreiche Manipulationen bietet auch das User-Menü in Bild 6. Im Color-Menü (Bild 7) ist wirklich jede Farbe schnell und einfach zu verändern.

Mit dem Character-Editor steht Ihnen ein äußerst leistungsfähiges Werkzeug zur Verfügung, mit dem man wirklich vernünftig arbeiten kann. In Bild 8 sehen Sie das Programm in voller Aktion. Sie werden es bei Ihrer weiteren Arbeit nicht mehr missen wollen! (Johannes Lauer/ef)


Bild 2. So meldet sich der

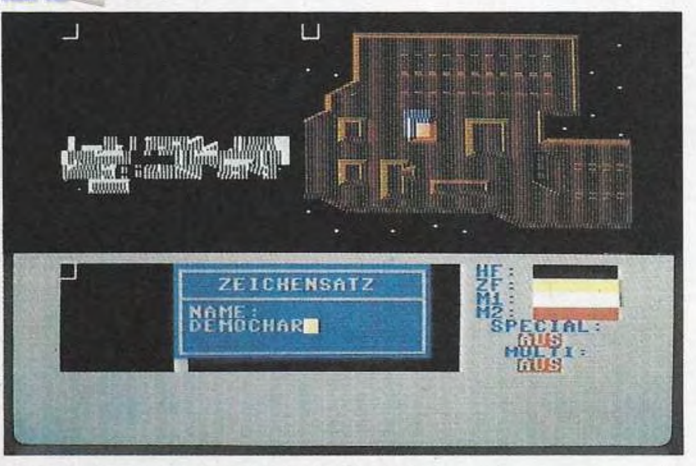


Bild 3. Ein Zeichensatz wird geladen

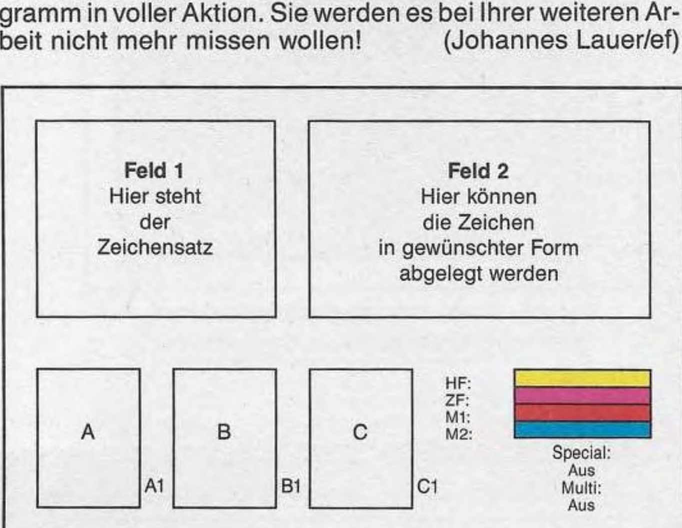


Bild 1. Bildschirmaufbau des »Character-Editors«

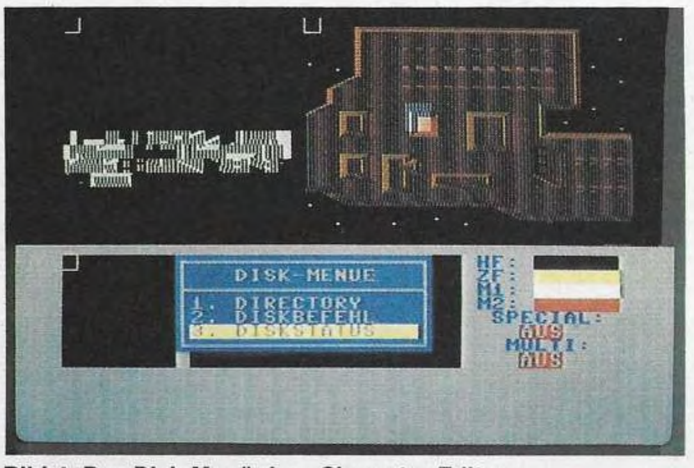
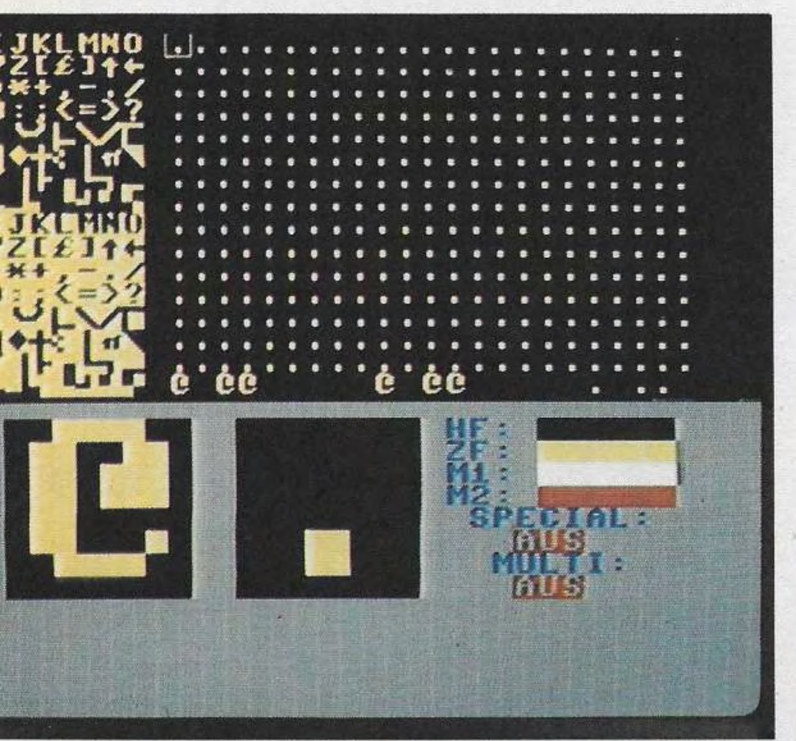


Bild 4. Das Disk-Menü des »Character-Editors«

Zeichensatz-



Editor

Erläuterungen

Zu V: (Verschieben)

Verschieben des Zeichens mit den Cursortasten.

Verlassen durch <←>, <RUN/STOP> oder <SPACE>.

Zu H: (Special an/aus)

Der Specialmodus erlaubt das Entwerfen von einfarbigen Zeichen im Multicolormodus.

Zu U: (User-Menü)

Im User-Menü gibt es drei Funktionen:
1. Copy-Zeichen
2. Clear-Zeichen
3. Invert-Zeichen

Mit dem Cursor kann man den Balken auf die gewünschte Funktion bewegen und mit <RETURN> die Funktion aufrufen, oder man drückt die entsprechende Zahl: <1> für Copy, <2> für Clear und <3> für Invert

Abbrechen kann man die Funktionen mit <←>, <RUN/STOP> oder <SPACE>.

Kurzinfo: Character-Editor

Programmart: Zeichensatz-Editor
Laden: LOAD "CHARACTER-EDITOR",8,1
Start: Nach dem Laden RUN eingeben
Steuerung: Die Bedienung des Editors erfolgt über die Tastatur. Bitte beachten Sie die Hinweise im Artikel.
Besonderheiten: Nach einem <RUN/STOP> <RESTORE> läßt sich das Programm mit RUN wieder starten, nach einem Reset mit SYS 2083.
Programmautor: Johannes Lauer

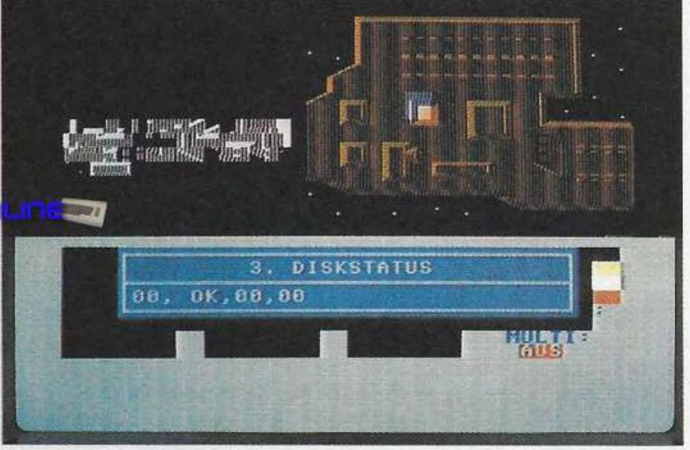


Bild 5. Resultat einer Statusabfrage

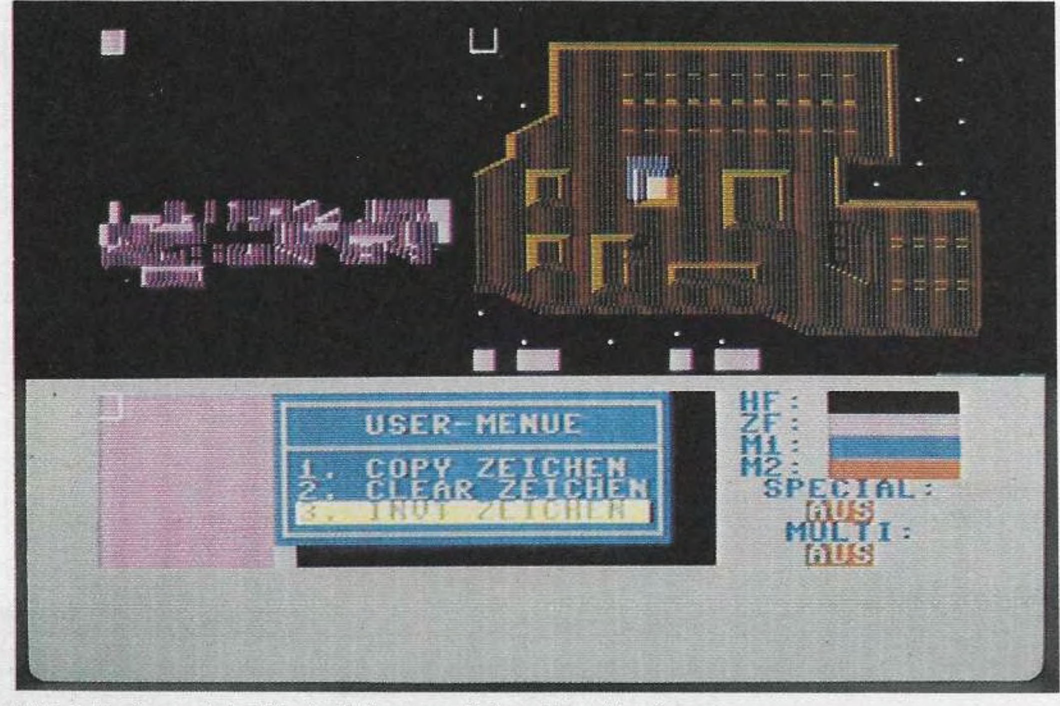


Bild 6. Umfangreiche Manipulationen mit dem User-Menü

Copy-Zeichen: (Zeichen kopieren)

Wenn diese Funktion angewählt wurde, kann man den Cursor in Feld 1 bewegen. Durch Drücken von <RETURN> wird der Anfang des zu kopierenden Bereichs gewählt. Jetzt erscheint über dem ersten Cursor ein zweiter, mit dem man das Ende des Bereichs festlegen kann. Darauf erscheint über dem zweiten Cursor ein dritter, mit dem man dann die Stelle festlegen kann, wohin kopiert werden soll.

Clear-Zeichen: (Zeichen löschen)

Funktioniert wie Copy-Zeichen, aber man braucht nur Anfang und Ende festzulegen.

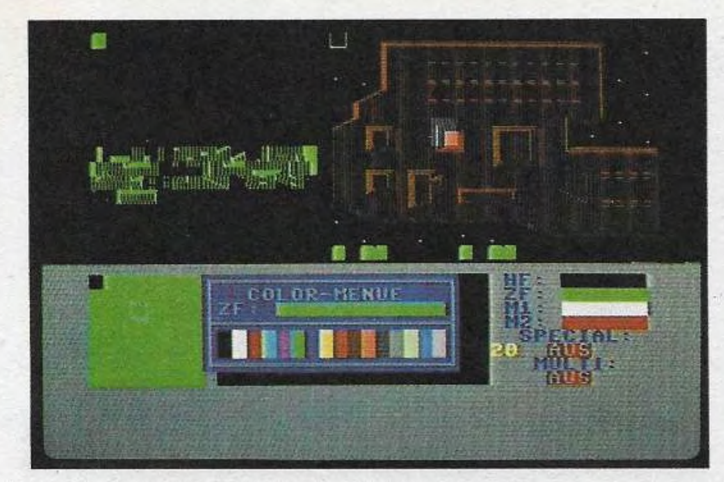


Bild 7. Farbenvielfalt durch das Color-Menü

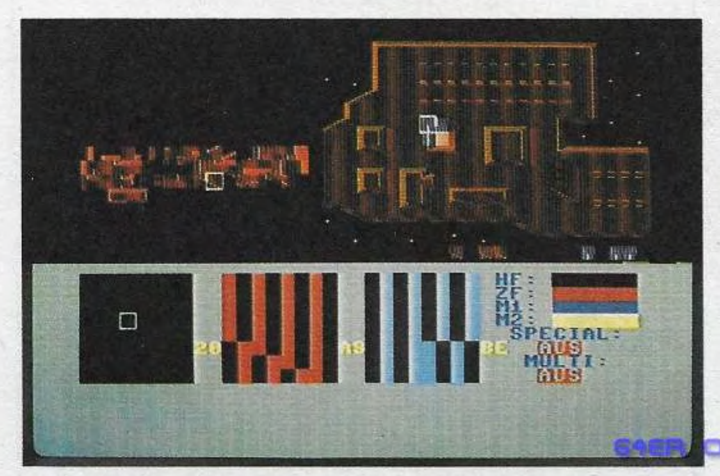


Bild 8. Das Programm in voller Aktion

Zu A:	Hier befindet sich das Editierfeld (auch Feld 3 genannt).
Zu A1:	Hier steht der Code des Zeichens in Hex-Darstellung. (Nur wenn die Anzeige angeschaltet wurde)
Zu B:	Hier sehen Sie das Zeichen, das unter dem Cursor in Feld 1 ist.
Zu B1:	Hier steht der Code des Zeichens in Hex-Darstellung. (Nur wenn die Anzeige angeschaltet wurde)
Zu C:	Hier ist das Zeichen abgebildet, das sich unter dem Cursor von Feld 2 befindet.
Zu C1:	Hier steht der Code des Zeichens in Hex-Darstellung. (Nur wenn die Anzeige angeschaltet wurde)
HF:	Hintergrundfarbe
ZF:	Zeichenfarbe
M1:	Multicolorfarbe 1
M2:	Multicolorfarbe 2
Special:	Anzeige des Special-Modus
Multi:	Anzeige des Multicolor-Modus

Tabelle 1. Kurzerläuterung der Felder aus Bild 1

Invert-Zeichen: (Zeichen invertieren)
Funktioniert wie Clear-Zeichen (siehe oben).

Zu S: (Save-Menü)
Funktioniert wie das User-Menü (siehe oben).
Funktionen:
1. Zeichensatz speichern
2. Zeichenfolgen speichern
3. Feld 2 speichern

Taste:	Funktion:	
Funktionen im 1. Feld:		
<HOME>	Cursor Home	
<CLR/HOME>	Zeichensatz löschen (mit Sicherheitsabfrage)	
<F3>	Nach Feld 2 wechseln	
<F5>	Nach Feld 3 wechseln	
Funktionen im 2. Feld:		
<HOME>	Cursor Home	
<CLR/HOME>	Feld 2 löschen (mit Sicherheitsabfrage)	
<F1>	Nach Feld 1 wechseln	
<F5>	Nach Feld 3 wechseln	
<INST>	Wie in Basic	
<DEL>	Wie in Basic	
<RETURN>	Zeichen von Feld 1 setzen	
<SHIFT RETURN>	Zeichen von Feld 3 setzen	
<CTRL 9>	RVS on	
<CTRL 0>	RVS off	
Alle anderen Tasten	Übernahme des Zeichens Beispiel:	
	<A>	Zeichen A
	<S>	Zeichen S
	<SHIFT S>	Herz
Funktionen im 3. Feld (Editierfeld):		
<*>	Zeile füllen	
<↑>	Spalte füllen	
<SHIFT *>	Zeile löschen	
<SHIFT ↑>	Spalte löschen	
<+>	Ein Zeichen vor	
<->	Ein Zeichen zurück	
<I>	Zeichen invertieren	
<F>	Zeichen füllen	
<Z>	Punkt setzen	
<X>	Punkt löschen	
<:>	Zeichen um X-Achse spiegeln	
<;>	Zeichen um Y-Achse spiegeln	
<V>	Verschieben	
<,>	Zeichen aus Feld 1 zum Bearbeiten in Feld 3 übernehmen	
<.>	Zeichen aus Feld 2 zum Bearbeiten in Feld 3 übernehmen	
<SHIFT ,>	Form des Zeichens aus Feld 1 in Feld 3 übernehmen	
<SHIFT .>	Form des Zeichens aus Feld 2 in Feld 3 übernehmen	
<0>	Mit Hintergrundfarbe arbeiten	
<1>	Mit Zeichenfarbe arbeiten	
<2>	Mit Multicolorfarbe 1 arbeiten	
<3>	Mit Multicolorfarbe 2 arbeiten	
<M>	Multicolor an/aus	
<H>	Special an/aus (funktioniert nur, wenn Multicolor eingeschaltet ist)	
<G>	Großschrift-Zeichensatz holen (mit Sicherheitsabfrage)	
<K>	Kleinschrift-Zeichensatz holen (mit Sicherheitsabfrage)	
<A>	Zeichencode-Anzeige an/aus	
<W>	Wechseln zwischen den beiden Zeichensätzen	
<U>	User-Menü	
<S>	Save-Menü	
<L>	Load-Menü	
<D>	Disk-Menü	
<C>	Color-Menü	
<SHIFT S>	Diskstatus	
<SHIFT D>	Directory	
<F1>	Nach Feld 1 wechseln	
<F3>	Nach Feld 2 wechseln	
<HOME>	Cursor Home	
<CLR/HOME>	Zeichen löschen	

Tabelle 2. Die Bedienung des »Character-Editors«

Nur bei Punkt 2 muß man noch Anfang und Ende des zu speichernden Bereichs festlegen.

Zu L: (Load-Menü)
Funktioniert wie das Save-Menü
1. Zeichensatz laden
2. Zeichenfolgen laden
3. Feld 2 laden

Bei Punkt 2 muß angegeben werden, wohin die Zeichenfolge geladen werden soll.

Zu D: (Disk-Menü)
In diesem Menü gibt es drei verschiedene Funktionen
1. Directory
2. Diskbefehl
3. Diskstatus

Zu C: (Color-Menü)
Mit den Cursortasten für aufwärts und abwärts wird der nächste Punkt angewählt und mit den Cursortasten für links und rechts die Farbe geändert.

```
Name : character-editor  0801 3307
-----------------------------------
0801 : 20 08 c3 07 9e 20 32 30  0b
0809 : 38 33 20 20 20 20 20 43  f1
0811 : 48 41 52 41 43 54 45 52  47
0819 : ab 45 44 49 a4 52 00 00  7e
0821 : 00 44 a9 08 20 a7 32 a9  0a
0829 : 07 8d 86 02 a9 00 8d 20  ea
0831 : d0 8d 21 d0 20 44 e5 a2  2b
0839 : 00 bd c3 08 c9 2a f0 07  c9
0841 : 9d fa 04 e8 4c 3a 08 e8  02
0849 : bd c3 08 c9 2a f0 07 9d  a5
0851 : 3c 05 e8 4c 49 08 e8 bd  c7
0859 : c3 08 c9 2a f0 07 9d 83  9d
0861 : 05 e8 4c 58 08 e8 bd c3  3f
0869 : 08 c9 2a f0 07 9d c6 05  81
0871 : e8 4c 67 08 a2 15 a0 1a  e4
0879 : bd aa 08 99 f9 d8 99 4f  2c
0881 : d9 99 9e d9 99 ec d9 88  83
0889 : d0 f1 20 96 08 b0 30 ca  89
0891 : d0 e4 b8 50 df a9 00 85  62
0899 : a2 a5 a2 c9 03 d0 fa 20  d2
08a1 : e4 ff c9 20 d0 02 38 60  ba
08a9 : 18 60 06 02 04 0c 05 03  6e
08b1 : 07 01 01 07 03 05 0c 04  eb
08b9 : 02 06 00 00 00 00 00 4c  57
08c1 : bf 32 03 08 01 12 01 03  06
08c9 : 14 05 12 2d 05 04 09 14  47
08d1 : 0f 12 2d 36 34 2a 09 0e  d0
08d9 : 20 31 39 38 36 2a 17 12  1c
08e1 : 09 14 14 05 0e 20 02 19  b6
08e9 : 2a 0a 0f 08 01 0e 0e 05  a0
08f1 : 13 20 0c 01 15 05 12 2a  4e
08f9 : a0 00 b1 fa 4a 08 0a 28  c2
0901 : 6a 91 fa c8 c0 08 d0 f2  81
0909 : 60 a0 00 b1 fa 0a 08 4a  a4
0911 : 28 2a 91 fa c8 c0 08 d0  66
0919 : f2 60 a0 00 b1 fa 48 c8  09
0921 : b1 fa 88 91 fa c8 c0 07  ab
0929 : d0 f5 68 91 fa 60 a0 07  83
0931 : b1 fa 48 88 b1 fa c8 91  bc
0939 : fa 88 d0 f7 68 91 fa 60  6a
0941 : a0 00 a2 08 a9 0d 20 d2  b4
0949 : ff b1 fa 85 02 a5 02 0a  fa
0951 : 85 02 b0 1c a9 01 85 c7  cf
0959 : ad f2 cf 8d 86 02 a9 20  84
0961 : 20 d2 ff a9 00 85 c7 ca  00
0969 : d0 e3 c8 c0 08 d0 d3 60  8c
0971 : a9 01 85 c7 ad f1 cf 8d  ba
0979 : 86 02 b8 50 e1 a0 00 a2  a1
0981 : 04 a9 0d 20 d2 ff b1 fa  8b
0989 : 85 02 a5 02 0a b0 24 0a  84
0991 : 85 02 b0 36 ad f2 cf 8d  d7
0999 : 86 02 a9 01 85 c7 a9 20  28
09a1 : 20 d2 ff 20 d2 ff a9 00  02
09a9 : 85 c7 ca d0 dd c8 c0 08  16
09b1 : d0 ed 60 0a 85 02 b0 09  fe
09b9 : ad 23 d0 8d 86 02 b8 50  da
09c1 : d9 ad f1 cf 8d 86 02 b8  6e
09c9 : 50 d0 ad 22 d0 8d 86 02  c9
09d1 : b8 50 c7 a9 24 85 fb a9  8a
09d9 : fb 85 bb a9 00 85 bc a9  2d
09e1 : 01 85 b7 a9 08 85 ba a9  b3
09e9 : 60 85 b9 20 d5 f3 a5 ba  87
09f1 : 20 b4 ff a5 b9 20 96 ff  17
09f9 : a9 00 85 90 a0 03 84 fb  42
0a01 : 20 a5 ff 85 fc a4 90 d0  7d
0a09 : 34 20 a5 ff a4 90 d0 2d  23
0a11 : a4 fb 88 d0 e9 a6 fc 20  f7
0a19 : cd bd a9 20 20 d2 ff 20  0c
0a21 : a5 ff a6 90 d0 17 aa f0  d4
0a29 : 06 20 d2 ff b8 50 f0 a9  19
0a31 : 0d 20 d2 ff ad 8d 02 d0  f4
0a39 : fb a0 02 d0 c1 20 42 f6  33
0a41 : 60 78 a9 00 85 5f 85 5a  66
0a49 : 85 58 a9 d0 85 60 a9 d8  32
0a51 : 85 5b a9 28 85 59 a9 33  23
0a59 : 85 01 20 bf a3 a9 37 85  ce
0a61 : 01 58 60 18 20 f0 ff a5  7e
0a69 : d1 85 fa a5 d2 85 fb 8c  d2
0a71 : 77 0a a5 fa 18 69 11 85  d2
0a79 : fa a5 fb 69 00 85 fb 60  4f
0a81 : 20 64 0a a5 fb 18 69 d4  da
0a89 : 85 fb a2 00 ad 34 03 a0  7e
0a91 : 00 91 fa c8 cc 35 03 d0  56
0a99 : f8 a5 fa 18 69 28 85 fa  09
0aa1 : a5 fb 69 00 85 fb e8 ec  54
0aa9 : 36 03 d0 e0 60 a9 00 85  0f
0ab1 : fa a9 04 85 fb a2 00 a0  48
0ab9 : 00 8a 91 fa e8 c8 c0 10  ba
0ac1 : d0 f7 a5 fa 18 69 28 85  ce
0ac9 : fa a5 fb 69 00 85 fb e0  a0
0ad1 : 00 d0 e4 60 20 64 0a a2  11
0ad9 : 00 ad 34 03 a0 00 91 fa  64
0ae1 : c8 cc 35 03 d0 f8 a5 fa  1e
```

Listing 1. »Character-Editor«, ein Zeichensatz-Editor der Spitzenklasse (bitte mit dem MSE, Seite 159, eingeben ▶













```
0ae9 : 18 69 28 85 fa a5 fb 69  10
0af1 : 00 85 fb e8 ec 36 03 d0  fe
0af9 : e0 60 78 a9 1a 8d 14 03  c1
0b01 : a9 0b 8d 15 03 a9 b3 8d  9d
0b09 : 12 d0 ad 11 d0 29 7f 8d  80
0b11 : 11 d0 a9 81 8d 1a d0 58  c2
0b19 : 60 ad 19 d0 8d 19 d0 29  e7
0b21 : 01 d0 07 ad 0d dc 58 4c  b4
0b29 : 54 0d ad 12 d0 c9 b3 b0  3d
0b31 : 36 ad f4 cf d0 0b ad 16  bd
0b39 : d0 29 ef 8d 16 d0 4c 4a  f9
0b41 : 0b ad 16 d0 09 10 8d 16  36
0b49 : d0 ad f2 cf 8d 21 d0 ad  27
0b51 : f3 cf 8d 20 d0 ad 18 d0  10
0b59 : 29 f0 09 08 8d 18 d0 a9  6e
0b61 : b3 8d 12 d0 4c 81 ea ad  51
0b69 : 16 d0 29 ef 8d 16 d0 a9  50
0b71 : 0c 8d 21 d0 8d 20 d0 ad  1f
0b79 : 18 d0 29 f0 09 04 8d 18  79
0b81 : d0 a9 00 8d 12 d0 4c 81  b3
0b89 : ea 78 a9 31 8d 14 03 a9  19
0b91 : ea 8d 15 03 a9 80 8d 1a  f1
0b99 : d0 a9 00 8d 21 d0 58 ad  45
0ba1 : 16 d0 29 ef 8d 16 d0 a9  88
0ba9 : 00 8d 20 d0 8d 21 d0 ad  12
0bb1 : 18 d0 29 f0 09 04 8d 18  b1
0bb9 : d0 60 a9 08 20 d2 ff a9  11
0bc1 : 8e 20 d2 ff 60 8d 34 03  5d
0bc9 : a9 10 8d 35 03 8d 36 03  00
0bd1 : a2 00 a0 00 4c 81 0a 8d  b0
0bd9 : 34 03 a9 17 8d 35 03 a9  be
0be1 : 0f 8d 36 03 a2 00 a0 11  74
0be9 : 4c d5 0a 8d 34 03 a9 17  84
0bf1 : 8d 35 03 a9 0f 8d 36 03  4b
0bf9 : a2 00 a0 11 4c 81 0a ff  df
0c01 : c0 00 80 40 00 80 40 00  ef
0c09 : 80 40 00 80 40 00 80 40  40
0c11 : 00 80 40 00 80 40 00 80  6d
0c19 : 40 00 ff c0 00 00 00 00  71
0c21 : 00 00 00 00 00 00 00 00  22
0c29 : 00 00 00 00 00 00 00 00  2a
0c31 : 00 00 00 00 00 00 00 00  32
0c39 : 00 00 00 00 00 00 00 ff  39
0c41 : ff c0 80 00 40 80 00 40  49
0c49 : 80 00 40 80 00 40 80 00  ee
0c51 : 40 80 00 40 80 00 40 80  e4
0c59 : 00 40 ff ff c0 00 00 00  85
0c61 : 00 00 00 00 00 00 00 00  62
0c69 : 00 00 00 00 00 00 00 00  6a
0c71 : 00 00 00 00 00 00 00 00  72
0c79 : 00 00 00 00 00 00 ad f0  12
0c81 : cf 85 fa a9 00 85 fb 06  2f
0c89 : fa 26 fb 06 fa 26 fb 06  33
0c91 : fa 26 fb a5 fb 18 69 20  b8
0c99 : 85 fb 60 a0 00 a2 08 a9  d1
0ca1 : 0d 20 2f 0d b1 fa 85 02  39
0ca9 : a5 02 0a 85 02 b0 1c a9  ec
0cb1 : 01 85 c7 ad f2 cf 8d 86  0d
0cb9 : 02 a9 20 20 d2 ff a9 00  70
0cc1 : 85 c7 ca d0 e3 c8 c0 08  8e
0cc9 : d0 d3 60 a9 01 85 c7 ad  87
0cd1 : f5 cf 8d 86 02 b8 50 e1  cd
0cd9 : a0 00 a2 04 a9 0d 20 2f  84
0ce1 : 0d b1 fa 85 02 a5 02 0a  a0
0ce9 : b0 24 0a 85 02 b0 36 ad  b8
0cf1 : f2 cf 8d 86 02 a9 01 85  7b
0cf9 : c7 a9 20 20 d2 ff 20 d2  f4
0d01 : ff a9 00 85 c7 ca d0 dd  57
0d09 : c8 c0 08 d0 cd 60 0a 85  61
0d11 : 02 b0 09 ad 23 d0 8d 86  5f
0d19 : 02 b8 50 d9 ad f5 cf 8d  ab
0d21 : 86 02 b8 50 d0 ad 22 d0  85
0d29 : 8d 86 02 b8 50 c7 20 40  d5
0d31 : 0d 98 18 69 11 aa a0 0a  bb
0d39 : 18 20 f0 ff 4c 4a 0d 8d  04
0d41 : ed cf 8e ee cf 8c ef cf  58
0d49 : 60 ad ed cf ae ee cf ac  f0
0d51 : ef cf 60 ad f6 cf f0 14  cf
0d59 : aa bd ab 08 ce f6 cf a2  f7
0d61 : 00 9d 27 d0 e8 e0 05 d0  5f
0d69 : f8 4c 31 ea a9 15 8d f6  98
0d71 : cf 4c 34 ea a2 00 86 5c  ce
0d79 : 86 5d a0 10 06 57 26 58  3c
0d81 : 26 5c 26 5d 38 a5 5c e5  f9
0d89 : 59 aa a5 5d e5 5a 90 06  cc
0d91 : 86 5c 85 5d e6 57 88 d0  3f
0d99 : e3 60 a9 d8 85 3f a9 03  31
0da1 : 85 40 20 8e 18 98 0a a8  e0
0da9 : a9 00 85 58 ad 10 d0 29  b0
0db1 : 01 f0 02 e6 58 b9 fe cf  77
0db9 : 38 e9 10 b0 02 c6 58 85  c3
0dc1 : 57 b9 ff cf 38 e9 29 85  71
0dc9 : 23 a9 08 85 59 a9 00 85  62
0dd1 : 5a 20 75 0d a5 57 85 24  ae
0dd9 : 46 23 46 23 46 23 a5 24  03
0de1 : 8d e8 0d a5 3f 18 69 0c  4d
0de9 : 85 3f a5 40 69 00 85 40  ac
0df1 : a5 3f 18 69 28 85 3f a5  60
0df9 : 40 69 00 85 40 c6 23 a5  b1
0e01 : 23 d0 ed a0 00 b1 3f 60  67
0e09 : a0 00 a2 08 a9 0d 20 9c  10
0e11 : 0e b1 fa 85 02 a5 02 0a  d1
0e19 : 85 02 b0 1c a9 01 85 c7  97
0e21 : ad f2 cf 8d 86 02 a9 20  4c
0e29 : 20 d2 ff a9 00 85 c7 ca  c8
0e31 : d0 e3 c8 c0 08 d0 d3 60  54
0e39 : a9 01 85 c7 ad fe cf 8d  ea
0e41 : 86 02 b8 50 e1 a0 00 a2  69
0e49 : 04 a9 0d 20 9c 0e b1 fa  60
0e51 : 85 02 a5 02 0a b0 24 0a  4c
0e59 : 85 02 b0 36 ad f2 cf 8d  9f
0e61 : 86 02 a9 01 85 c7 a9 20  f0
0e69 : 20 d2 ff 20 d2 ff a9 00  ca
0e71 : 85 c7 ca d0 dd c8 c0 08  de
0e79 : d0 cd 60 0a 85 02 b0 09  c6
0e81 : ad 23 d0 8d 86 02 b8 50  a2
0e89 : d9 ad fe cf 8d 86 02 b8  79
0e91 : 50 d0 ad 22 d0 8d 86 02  91
0e99 : b8 50 c7 20 40 0d 98 18  6e
0ea1 : 69 11 aa a0 14 18 20 f0  b6
0ea9 : ff 4c 4a 0d a2 10 a0 00  30
0eb1 : 18 20 f0 ff 20 7f 0c ad  9f
0eb9 : f4 cf d0 03 4c 41 09 ad  78
0ec1 : fb cf d0 f8 ea ea ea ea  7e
0ec9 : ea 4c 7e 09 a9 30 8d 54  95
0ed1 : 0a 4c 42 0a a9 28 8d 54  8e
0ed9 : 0a 4c 42 0a a0 01 20 9b  a5
0ee1 : 0d 8d ff cf 20 82 0c ad  50
0ee9 : f4 cf d0 11 a5 40 18 69  ab
0ef1 : d4 85 40 a0 00 b1 3f 8d  52
0ef9 : f5 cf 4c 9c 0c 4c 5c 13  37
0f01 : ea ea ea ea ea ea ea ea  00
0f09 : ea ea ea ea ea ea ea a0  73
0f11 : 02 20 9b 0d 8d fd cf 20  f4
0f19 : 82 0c ad f4 cf d0 11 a5  be
0f21 : 40 18 69 d4 85 40 a0 00  3f
0f29 : b1 3f 8d fe cf 4c 09 0e  5d
0f31 : 4c a2 13 ea ea ea ea ea  78
0f39 : ea ea ea ea ea ea ea ea  38
0f41 : ea ea 53 50 45 43 49 41  95
0f49 : 4c 3a 00 4d 55 4c 54 49  f8
0f51 : 3a 00 1c 12 41 4e 20 00  dc
0f59 : 1c 12 41 55 53 92 00 a9  96
0f61 : 06 8d 86 02 a0 1e a2 15  bf
0f69 : 18 20 f0 ff a9 43 a0 0f  23
0f71 : 20 1e ab a2 16 a0 20 18  f7
0f79 : 20 f0 ff ad fb cf d0 08  58
0f81 : a9 59 a0 0f 20 1e ab 60  43
0f89 : a9 53 a0 0f 20 1e ab 60  48
0f91 : a9 06 20 04 41 a2 17 a0  8d
0f99 : 1f 18 20 f0 ff a9 4c a0  aa
0fa1 : 0f 20 1e ab a2 18 a0 20  6b
0fa9 : 18 20 f0 ff ad f4 cf d0  71
0fb1 : d7 4c 81 0f 1f 48 46 3a  b2
0fb9 : 00 1f 5a 46 3a 00 1f 4d  63
0fc1 : 31 3a 00 1f 4d 32 3a 00  43
0fc9 : a0 1d a2 11 18 20 f0 ff  09
0fd1 : a9 b5 a0 0f 20 1e ab ad  5c
0fd9 : f2 cf 8d 86 02 a9 20 20  15
0fe1 : d2 ff a2 06 a9 01 85 c7  64
0fe9 : a9 20 20 40 0d 20 d2 ff  d0
0ff1 : 20 4a 0d ca d0 f2 a9 00  1e
0ff9 : 85 c7 60 a0 1d a2 12 18  ed
1001 : 20 f0 ff a9 ba a0 0f 20  fb
1009 : 1e ab ad f1 cf 4c db 0f  93
1011 : a0 1d a2 13 18 20 f0 ff  91
1019 : a9 bf a0 0f 20 1e ab ad  a9
1021 : 22 d0 4c db 0f a0 1d a2  ea
1029 : 14 18 20 f0 ff a9 c4 a0  11
1031 : 0f 20 1e ab ad 23 d0 4c  1d
1039 : db 0f 20 c9 0f 20 fc 0f  e1
1041 : 20 11 10 4c 26 10 20 60  9c
1049 : 0f 4c 91 0f 20 3b 10 4c  79
1051 : 47 10 20 7f 0c a4 cb ad  09
1059 : 8d 02 20 40 0d 29 01 d0  b7
1061 : 29 c0 02 d0 1d ad f4 cf  38
1069 : d0 03 4c f9 08 ad fb cf  8b
1071 : d0 f8 ad 12 d0 d0 fb 20  2f
1079 : f9 08 4c f9 08 ea ea ea  22
1081 : ea ea c0 07 d0 03 4c 2f  a6
1089 : 09 60 c0 02 d0 1d ad f4  c9
1091 : cf d0 03 4c 0a 09 ad fb  aa
1099 : cf d0 f8 ad 12 d0 d0 fb  a7
10a1 : 20 0a 09 4c 0a 09 ea ea  fd
10a9 : ea ea ea c0 07 d0 03 4c  77
10b1 : 1b 09 60 20 53 10 20 4a  38
10b9 : 0d c0 39 f0 f5 c0 3f f0  d7
10c1 : f1 c0 3c f0 ed 20 3f 13  42
10c9 : a2 13 a0 14 88 d0 fd ca  3c
10d1 : d0 f8 4c b4 10 8a 0a aa  9a
10d9 : a0 00 bd fe cf c9 ff f0  f6
10e1 : 19 fe fe cf 20 40 0d a2  b0
10e9 : 31 a0 21 88 d0 fd ca d0  8d
10f1 : f8 20 4a 0d c8 c0 08 d0  82
10f9 : e1 60 8a 4a 8d 04 11 ad  8f
1101 : 10 d0 09 02 8d 10 d0 fe  97
1109 : fe cf 4c e5 10 8a 0a aa  91
1111 : a0 00 bd fe cf c9 00 f0  2e
1119 : 19 de fe cf 20 40 0d a2  d8
1121 : 31 a0 21 88 d0 fd ca d0  c5
1129 : f8 20 4a 0d c8 c0 08 d0  ba
1131 : e1 60 8a 4a 49 ff 8d 3e  75
1139 : 11 ad 10 d0 29 fd 8d 10  18
1141 : d0 de fe cf 4c 1d 11 8a  41
1149 : 0a aa a0 00 fe ff cf 20  40
1151 : 40 0d a2 32 a0 21 88 d0  de
1159 : fd ca d0 f8 20 4a 0d c8  29
1161 : c0 08 d0 e8 60 8a 0a aa  4e
1169 : a0 00 de ff cf 20 40 0d  da
1171 : a2 32 a0 21 88 d0 fd ca  15
1179 : d0 f8 20 4a 0d c8 c0 08  41
1181 : d0 e8 60 a4 cb ad 8d 02  d6
1189 : 20 40 0d 29 01 d0 28 c0  eb
1191 : 02 d0 10 ad e0 cf c9 0f  87
1199 : f0 09 a2 01 20 d6 10 ee  ad
11a1 : e0 cf 60 c0 07 d0 fb ad  db
```

```
11a9 : e1 cf c9 0f f0 f4 a2 01  09
11b1 : 20 48 11 ee e1 cf 60 c0  b7
11b9 : 02 d0 10 ad e0 cf c9 00  91
11c1 : f0 09 a2 01 20 0e 11 ce  53
11c9 : e0 cf 60 c0 07 d0 fb ad  03
11d1 : e1 cf c9 00 f0 f4 a2 01  4f
11d9 : 20 66 11 ce e1 cf 60 20  a9
11e1 : 84 11 20 4a 0d 20 dd 0e  a5
11e9 : 20 fb 12 20 e4 ff c9 13  2b
11f1 : d0 03 4c fd 17 c9 93 d0  c5
11f9 : 03 4c 0e 1d 60 20 e4 ff  e4
1201 : a4 cb ad 8d 02 29 01 d0  b7
1209 : 27 98 c9 04 d0 06 a9 01  55
1211 : 8d fc cf 60 c9 05 d0 06  30
1219 : a9 02 8d fc cf 60 c9 06  f9
1221 : d0 06 a9 03 8d fc cf 60  80
1229 : c9 03 d0 03 ea ea ea 60  7b
1231 : 98 c9 04 d0 06 a0 01 20  73
1239 : 5b 12 ea c9 05 d0 06 a0  c1
1241 : 02 20 5b 12 ea c9 06 d0  23
1249 : 06 ea ea ea ea ea ea c9  21
1251 : 03 d0 06 ea ea ea ea ea  23
1259 : ea 60 ee e6 cf ad e6 cf  b1
1261 : c9 50 f0 01 60 ee e7 cf  6b
1269 : a9 00 8d e6 cf ad e7 cf  fc
1271 : c9 14 d0 f0 a9 00 8d e7  37
1279 : cf c0 02 d0 04 ee f2 cf  66
1281 : 60 ee f3 cf 60 a4 cb ad  05
1289 : 8d 02 20 40 0d 29 01 d0  e7
1291 : 28 c0 02 d0 10 ad e2 cf  4e
1299 : c9 16 f0 09 a2 02 20 d6  33
12a1 : 10 ee e2 cf 60 c0 07 d0  a5
12a9 : fb ad e3 cf c9 0e f0 f4  28
12b1 : a2 02 20 48 11 ee e3 cf  1d
12b9 : 60 c0 02 d0 10 ad e2 cf  ae
12c1 : c9 00 f0 09 a2 02 20 0e  be
12c9 : 11 ce e2 cf 60 c0 07 d0  be
12d1 : fb ad e3 cf c9 00 f0 f4  e0
12d9 : a2 02 20 66 11 ce e3 cf  08
12e1 : 60 ad f0 cf 8d 69 06 8d  a5
12e9 : 6b 06 8d 6c 06 20 48 13  f1
12f1 : 8d 69 da 8d 6b da 8d 6c  38
12f9 : da 60 ad ff cf 8d 72 06  ae
1301 : 8d 74 06 8d 75 06 20 88  15
1309 : 13 8d 72 da 8d 74 da 8d  de
1311 : 75 da 60 ad fd cf 8d 7b  4d
1319 : 06 8d 7d 06 8d 7e 06 20  2b
1321 : ce 13 8d 7b da 8d 7d da  11
1329 : 8d 7e da 60 20 ad 0e 4c  f8
1331 : e2 12 20 dd 0e 4c fb 12  37
1339 : 20 10 0f 4c 14 13 20 2d  63
1341 : 13 20 33 13 4c 39 13 ad  ca
1349 : f4 cf d0 04 ad f1 cf 60  44
1351 : ad fb cf d0 f7 ad f1 cf  5e
1359 : 09 08 60 a5 40 18 69 d4  47
1361 : 85 40 a0 00 b1 3f 48 29  b7
1369 : 08 f0 0e a9 00 8d f7 cf  8e
1371 : 68 29 f7 8d f5 cf 4c d9  e0
1379 : 0c a9 ff 8d f7 cf 68 29  fd
1381 : f7 8d f5 cf 4c 9c 0c ad  eb
1389 : f4 cf d0 04 ad f5 cf 60  a4
1391 : ad f7 cf d0 06 ad f5 cf  8d
1399 : 09 08 60 ad f5 cf 29 d7  a6
13a1 : 60 a5 40 18 69 d4 85 40  bb
13a9 : a0 00 b1 3f 48 29 08 f0  6d
13b1 : 0e a9 00 8d f8 cf 68 29  48
13b9 : f7 8d fe cf 4c 46 0e a9  b3
13c1 : ff 8d f8 cf 68 29 f7 8d  8a
13c9 : fe cf 4c 09 0e ad f4 cf  a5
13d1 : d0 04 ad fe cf 60 ad f8  97
13d9 : cf d0 06 ad fe cf 09 08  ea
13e1 : 60 ad fe cf 29 d7 60 c0  26
13e9 : 30 0c 03 00 ff 55 aa 80  37
13f1 : 40 20 10 08 04 02 01 20  db
13f9 : e4 ff a4 cb ad 8d 02 29  21
1401 : 01 d0 28 c0 02 d0 10 ad  cf
1409 : e4 cf c9 07 f0 08 a2 03  08
1411 : 20 d6 10 ee e4 cf 60 c0  4e
1419 : 07 d0 fb ad e5 cf c9 07  4f
1421 : f0 f4 a2 03 20 48 11 ee  fb
1429 : e5 cf 60 c0 02 d0 10 ad  68
1431 : e4 cf c9 00 f0 08 a2 03  4f
1439 : 20 0e 11 ce e4 cf 60 c0  4e
1441 : 07 d0 fb ad e5 cf c9 00  69
1449 : f0 f4 a2 03 20 66 11 ce  d3
1451 : e5 cf 60 20 e4 ff a4 cb  b2
1459 : ad 8d 02 29 01 d0 2d c0  3f
1461 : 02 d0 15 ad e4 cf c9 03  c0
1469 : f0 0d a2 03 20 d6 10 a2  27
1471 : 03 20 d6 10 ee e4 cf 60  52
1479 : c0 07 d0 fb ad e5 cf c9  4d
1481 : 07 f0 f4 a2 03 20 48 11  06
1489 : ee e5 cf 60 c0 02 d0 15  f3
1491 : ad e4 cf c9 00 f0 0d a2  de
1499 : 03 20 0e 11 a2 03 20 0e  31
14a1 : 11 ce e4 cf 60 c0 07 d0  16
14a9 : fb ad e5 cf c9 00 f0 f4  38
14b1 : a2 03 20 66 11 ce e5 cf  68
14b9 : 60 20 7f 0c ea ea ae e4  15
14c1 : cf bd f0 13 8d ce 14 ac  06
14c9 : e5 cf b1 fa 09 20 91 fa  2f
14d1 : 60 20 7f 0c ea ea ae e4  2d
14d9 : cf bd f0 13 49 ff 8d e8  c2
14e1 : 14 ac e5 cf b1 fa 29 7f  55
14e9 : 91 fa 60 20 7f 0c ea ea  ed
14f1 : ae e4 cf ac e5 cf bd e8  40
14f9 : 13 48 49 ff 31 fa 91 fa  aa
1501 : ae e8 cf 68 3d ec 13 11  ce
1509 : fa 91 fa 60 ad e8 cf 8d  13
1511 : e9 cf a9 00 8d e8 cf 20  ec
1519 : ec 14 ad e9 cf 8d e8 cf  64
1521 : 60 ea ea ea a5 cb c9 23  34
1529 : d0 05 a0 00 8c e8 cf c9  87
1531 : 38 d0 05 a0 01 8c e8 cf  de
1539 : c9 3b d0 05 a0 02 8c e8  93
1541 : cf c9 08 d0 05 a0 03 8c  8b
1549 : e8 cf 60 ad f4 cf d0 14  20
1551 : 20 f8 13 a5 cb c9 0c d0  44
1559 : 03 4c ef 18 c9 17 d0 03  20
1561 : 4c f5 18 60 ad fb cf d0  56
1569 : e7 20 59 14 a5 cb c9 0c  31
1571 : d0 03 4c fb 18 c9 17 d0  23
1579 : 03 4c 01 19 60 20 4c 15  68
1581 : 20 ad 0e 20 25 15 20 fe  79
1589 : 11 20 ab 41 60 ea ea 60  87
1591 : 20 7f 0c a9 00 8d f9 cf  9d
1599 : a9 07 8d fa cf ac f9 cf  72
15a1 : b1 fa 48 ac fa cf b1 fa  62
15a9 : ac f9 cf 91 fa ac fa cf  18
15b1 : 68 91 fa ee f9 cf ce fa  cd
15b9 : cf ad f9 cf c9 04 d0 dd  93
15c1 : 60 40 80 10 20 04 08 01  a8
15c9 : 02 a2 00 a5 22 4a 85 22  a0
15d1 : 90 1a bd c2 15 9d 34 03  4b
15d9 : e8 e0 08 d0 ee ad 34 03  81
15e1 : a2 00 18 7d 35 03 e8 e0  0a
15e9 : 07 d0 f7 60 a9 00 b8 50  80
15f1 : e4 a2 00 a5 22 4a 85 22  aa
15f9 : 90 1a bd f0 13 9d 34 03  19
1601 : e8 e0 08 d0 ee ad 34 03  a9
1609 : a2 00 18 7d 35 03 e8 e0  32
1611 : 07 d0 f7 60 a9 00 b8 50  a8
1619 : e4 20 7f 0c ad f4 cf d0  d2
1621 : 11 a0 00 b1 fa 85 22 20  5d
1629 : f2 15 91 fa c8 c0 08 d0  be
1631 : f2 60 ad fb cf d0 ea a0  ae
1639 : 00 b1 fa 85 22 20 ca 15  fa
1641 : 91 fa c8 c0 08 d0 f2 60  2d
1649 : 20 7f 0c ac e5 cf a9 00  45
1651 : 91 fa 60 20 7f 0c ad f4  74
1659 : cf d0 19 ae e4 cf bd f0  52
1661 : 13 49 ff 8d 6c 16 a0 00  c4
1669 : b1 fa 29 fe 91 fa c8 c0  57
1671 : 08 d0 f5 60 ad fb cf d0  06
1679 : e2 ae e4 cf bd e8 13 49  e7
1681 : ff 8d 8a 16 a0 00 b1 fa  73
1689 : 29 fc 91 fa c8 c0 08 d0  48
1691 : f5 60 20 7f 0c a0 00 98  a5
1699 : 91 fa c8 c0 08 d0 f9 60  a1
16a1 : 20 7f 0c a0 00 b1 fa 49  a4
16a9 : ff 91 fa c8 c0 08 d0 f5  c4
16b1 : 60 20 7f 0c ad f4 cf d0  e6
16b9 : 08 ac e5 cf a9 ff 91 fa  61
16c1 : 60 ad fb cf d0 f3 ac e5  1c
16c9 : cf ae e8 cf bd ec 13 91  d6
16d1 : fa 60 20 7f 0c ad f4 cf  95
16d9 : d0 17 ae e4 cf bd f0 13  52
16e1 : 8d e9 16 a0 00 b1 fa 09  88
16e9 : 01 91 fa c8 c0 08 d0 f5  06
16f1 : 60 ad fb cf d0 e4 20 7a  ca
16f9 : 16 ae e4 cf bd e8 13 ae  66
1701 : e8 cf 3d ec 13 8d 0e 17  c2
1709 : a0 00 b1 fa 09 02 91 fa  52
1711 : c8 c0 08 d0 f5 60 20 7f  37
1719 : 0c ad f4 cf d0 0c a0 00  23
1721 : a9 ff 91 fa c8 c0 08 d0  e2
1729 : f9 60 ad fb cf d0 ef ae  de
1731 : e8 cf bd ec 13 a0 00 91  67
1739 : fa c8 c0 08 d0 f9 60 ad  82
1741 : e5 cf c9 00 f0 0b ce e5  ef
1749 : cf a2 03 20 66 11 4c 40  cf
1751 : 17 ad fa 07 c9 31 f0 10  e8
1759 : ad e4 cf f0 22 ce e4 cf  56
1761 : a2 03 20 0e 11 4c 59 17  56
1769 : ad e4 cf c9 00 f0 10 ce  1b
1771 : e4 cf a2 03 20 0e 11 a2  42
1779 : 03 20 0e 11 4c 69 17 60  5f
1781 : c9 20 b0 07 a9 20 ea ea  84
1789 : b8 50 44 c9 40 b0 03 b8  bb
1791 : 50 3d c9 60 b0 06 38 e9  ee
1799 : 40 b8 50 33 c9 80 b0 06  1f
17a1 : 38 e9 20 b8 50 29 c9 a0  a4
17a9 : b0 05 a9 20 b8 50 20 c9  6c
17b1 : c0 b0 06 38 e9 40 b8 50  76
17b9 : 16 c9 e0 b0 06 38 e9 60  8d
17c1 : b8 50 bd c9 ff b0 06 38  58
17c9 : e9 40 b8 50 b3 a9 5e a6  5a
17d1 : c7 f0 03 18 69 80 60 ad  4c
17d9 : e3 cf c9 00 f0 0b ce e3  81
17e1 : cf a2 02 20 66 11 4c d8  58
17e9 : 17 ad e2 cf c9 00 f0 0b  00
17f1 : ce e2 cf a2 02 20 0e 11  f4
17f9 : 4c ea 17 60 ad e1 cf c9  49
1801 : 00 f0 0b ce e1 cf a2 01  3f
1809 : 20 66 11 4c fd 17 ad e0  3b
1811 : cf c9 00 f0 0b ce e0 cf  2d
1819 : a2 01 20 0e 11 4c 0f 18  e6
1821 : 60 20 e4 ff c9 0d d0 03  19
1829 : 4c f2 1d c9 8d d0 03 4c  73
1831 : 01 1e c9 13 d0 03 4c d8  1e
1839 : 17 c9 93 d0 03 4c 61 1d  86
1841 : c9 14 d0 03 4c 10 1e c9  fa
```

Listing 1. »Character-Editor« (Fortsetzung)

```
1849 : 94 d0 03 4c 6c 1e c9 86   7c
1851 : d0 01 60 a2 85 8e 5c 18   7d
1859 : a2 00 c9 87 f0 f4 e8 ee   97
1861 : 5c 18 e0 07 d0 f4 c9 00   be
1869 : d0 01 60 ea ea ea ea c9   74
1871 : 03 f0 df c9 83 f0 db c9   e0
1879 : 9d f0 d7 c9 11 f0 d3 c9   39
1881 : 91 f0 cf c9 1d f0 cb a2   85
1889 : 00 8e ea cf 60 b9 94 18   db
1891 : 8d b1 0d 60 01 02 04 08   87
1899 : 10 20 40 80 20 86 12 20   98
18a1 : 39 13 a9 ff 8d ea cf 20   7e
18a9 : 22 18 8d eb cf 20 ba 1d   db
18b1 : f0 01 60 a0 02 20 9b 0d   f8
18b9 : a0 00 ad eb cf 20 e3 1d   0a
18c1 : 91 3f a5 40 18 69 d4 85   8e
18c9 : 40 ad f4 cf d0 06 ad f1   ef
18d1 : cf 91 3f 60 ad fb cf d0   e0
18d9 : f5 ad f1 cf 09 08 91 3f   b1
18e1 : 60 ad ff cf 8d f0 cf 20   f1
18e9 : a9 42 20 4c 43 60 20 ba   72
18f1 : 14 4c 3f 13 20 d2 14 4c   df
18f9 : 3f 13 20 ec 14 4c 3f 13   2e
1901 : 20 0d 15 4c 3f 13 ad fd   b6
1909 : cf 8d f0 cf 20 b8 42 20   e6
1911 : 4c 43 60 ee f0 cf 4c 3f   32
1919 : 13 ce f0 cf 4c 3f 13 20   15
1921 : 91 15 4c 3f 13 20 1a 16   ff
1929 : 4c 3f 13 20 40 17 20 ee   f9
1931 : 1c f0 0d a9 00 8d f4 cf   1e
1939 : 8d fb cf a9 30 4c 4b 19   b2
1941 : 8d fb cf a9 ff 8d f4 cf   d5
1949 : a9 31 8d fa 07 a5 fa 48   68
1951 : a5 fb 48 a9 00 85 fa a9   a7
1959 : d8 85 fb a2 00 a0 00 b1   af
1961 : fa 09 08 91 fa c8 c0 10   2d
1969 : d0 f5 a5 fa 18 69 28 85   75
1971 : fa 90 02 e6 fb e8 e0 10   bb
1979 : d0 e3 68 85 fb 68 85 fa   15
1981 : 20 4d 10 4c 3f 13 a5 fb   f1
1989 : 48 a5 fa 48 a9 80 85 fa   16
1991 : a9 06 85 fb a9 00 85 fc   c9
1999 : a9 c0 85 fd a0 00 b1 fa   8a
19a1 : 91 fc a5 fa 18 69 01 85   55
19a9 : fa 90 02 e6 fb a5 fc 18   5a
19b1 : 69 01 85 fc 90 02 e6 fd   4c
19b9 : a5 fb c9 07 d0 de a5 fa   40
19c1 : c9 e8 d0 d8 a9 80 85 fa   f8
19c9 : a9 da 85 fb a9 68 85 fc   ae
19d1 : a9 c1 85 fd a0 00 b1 fa   43
19d9 : 91 fc a5 fa 18 69 01 85   8d
19e1 : fa 90 02 e6 fb a5 fc 18   92
19e9 : 69 01 85 fc 90 02 e6 fd   84
19f1 : a5 fb c9 db d0 de a5 fa   12
19f9 : c9 e8 d0 d8 68 85 fa 68   f5
1a01 : 85 fb 60 ad a0 19 ae a2   24
1a09 : 19 8e a0 19 8d a2 19 ad   62
1a11 : d8 19 ae da 19 8e d8 19   18
1a19 : 8d da 19 20 87 19 ad a0   97
1a21 : 19 ae a2 19 8e a0 19 8d   ca
1a29 : a2 19 ad d8 19 ae da 19   83
1a31 : 8e d8 19 8d da 19 60 f0   fd
1a39 : c0 c0 c0 c0 c0 c0 c0 c0   38
1a41 : c0 c0 c0 c0 c0 c0 c0 c0   40
1a49 : ee dd a0 a0 83 8f 8c 8f   68
1a51 : 92 ad 8d 85 8e 95 85 a0   bb
1a59 : a0 a0 dd dd a0 a0 a0 a0   4f
1a61 : a0 a0 a0 a0 a0 a0 a0 a0   60
1a69 : a0 a0 a0 a0 dd eb c0 c0   57
1a71 : c0 c0 c0 c0 c0 c0 c0 c0   70
1a79 : c0 c0 c0 c0 c0 c0 f3 ed   9f
1a81 : c0 c0 c0 c0 c0 c0 c0 c0   80
```

```
1a89 : c0 c0 c0 c0 c0 c0 c0 c0   88
1a91 : fd 20 87 19 a2 00 bd 38   35
1a99 : 1a 9d b0 06 a9 06 9d b0   11
1aa1 : da e8 e0 12 d0 f0 a2 00   89
1aa9 : bd 4a 1a 9d d8 06 a9 06   36
1ab1 : 9d d8 da e8 e0 12 d0 f0   52
1ab9 : a2 00 bd 5c 1a 9d 00 07   f3
1ac1 : a9 06 9d 00 db e8 e0 12   81
1ac9 : d0 f0 a2 00 bd 6e 1a 9d   ad
1ad1 : 28 07 a9 06 9d 28 db e8   04
1ad9 : e0 12 d0 f0 a2 00 bd 5c   ee
1ae1 : 1a 9d 50 07 9d 78 07 a9   cc
1ae9 : 06 9d 50 db 9d 78 db e8   2c
1af1 : e0 12 d0 ea a2 00 bd 80   8e
1af9 : 1a 9d a0 07 a9 06 9d a0   6d
1b01 : db e8 e0 12 d0 f0 a2 00   ea
1b09 : 8a 9d 51 db 9d 79 db e8   18
1b11 : e0 10 d0 f4 a9 a0 8d ea   78
1b19 : 06 8d 12 07 8d 3a 07 8d   2d
1b21 : 62 07 8d 8a 07 8d b2 07   71
1b29 : a9 00 8d ea da 8d 12 db   ad
1b31 : 8d 3a db 8d 62 db 8d 8a   d4
1b39 : db 8d b2 db a2 00 a9 a0   15
1b41 : 9d c9 07 a9 00 9d c9 db   86
1b49 : e8 e0 12 d0 f1 60 a2 13   13
1b51 : a0 09 20 0c e5 a9 01 85   ba
1b59 : c7 a9 ba a0 0f 20 1e ab   79
1b61 : a9 00 85 c7 60 a9 ba 8d   be
1b69 : 5b 1b 4c 4f 1b a9 bf 8d   68
1b71 : 5b 1b 4c 4f 1b a9 c4 8d   84
1b79 : 5b 1b 4c 4f 1b a9 b5 8d   50
1b81 : 5b 1b 4c 4f 1b 20 7e 1b   4a
1b89 : a9 92 8d 01 07 60 66 6e   e9
1b91 : 76 7e 86 ae ec cf bd 8f   21
1b99 : 1b 8d 9e 1b 4c 66 1b 07   f8
1ba1 : ad ec cf d0 06 ae f1 cf   0f
1ba9 : 8e a0 1b c9 01 d0 06 ae   93
1bb1 : 22 d0 8e a0 1b c9 02 d0   9d
1bb9 : 06 ae 23 d0 8e a0 1b c9   e7
1bc1 : 03 d0 06 ae f2 cf 8e a0   ad
1bc9 : 1b c9 04 d0 06 ae f3 cf   29
1bd1 : 8e a0 1b a2 00 ad a0 1b   f1
1bd9 : 9d 05 db e8 e0 0c d0 f8   b0
1be1 : 60 ad ec cf d0 03 20 2c   4b
1be9 : 1c c9 01 d0 03 ee 22 d0   16
1bf1 : c9 02 d0 03 ee 23 d0 c9   2f
1bf9 : 03 d0 03 ee f2 cf c9 04   e0
1c01 : d0 03 ee f3 cf 60 ad ec   1d
1c09 : cf d0 03 20 43 1c c9 01   43
1c11 : d0 03 ce 22 d0 c9 02 d0   60
1c19 : 03 ce 23 d0 c9 03 d0 03   64
1c21 : ce f2 cf c9 04 d0 03 ce   06
1c29 : f3 cf 60 ee f1 cf ad f4   38
1c31 : cf f0 0c ad f1 cf c9 08   06
1c39 : d0 05 a9 00 8d f1 cf a9   f1
1c41 : 00 60 ce f1 cf ad f4 cf   41
1c49 : f0 0c ad f1 cf c9 ff d0   d6
1c51 : 05 a9 07 8d f1 cf a9 00   e3
1c59 : 60 60 a9 00 8d ec cf 20   13
1c61 : 92 1a 20 a1 1b 20 94 1b   78
1c69 : a4 cb ad 8d 02 29 01 d0   1f
1c71 : 3d c0 02 d0 09 20 e2 1b   fc
1c79 : 20 a1 1b 4c 92 1c c0 07   d5
1c81 : d0 0f ee ec cf ad ec cf   f0
1c89 : c9 05 d0 05 a9 00 8d ec   54
1c91 : cf 4c a4 3d ea 20 04 1a   4b
1c99 : 4c d2 1c 20 4d 10 20 a1   73
1ca1 : 1b a2 50 a0 f0 88 d0 fd   c8
1ca9 : ca d0 f8 4c 66 1c c0 02   f1
1cb1 : d0 09 20 07 1c 20 a1 1b   6e
1cb9 : 4c 92 1c c0 07 d0 0f ce   3e
1cc1 : ec cf ad ec cf c9 ff d0   8b
```

```
1cc9 : 05 a9 04 8d ec cf 4c 92   f9
1cd1 : 1c 20 4d 10 ad f4 cf f0   f6
1cd9 : 08 20 ee 43 4c e5 1c ea   50
1ce1 : ea ad f1 cf 20 c6 0b 4c   15
1ce9 : 3f 13 ea ea ea ad f1 cf   4d
1cf1 : 29 07 8d f1 cf ad f4 cf   1d
1cf9 : 60 90 4c 4f 45 53 43 48   2b
1d01 : 45 4e 20 3f 00 20 43 4f   0a
1d09 : 44 45 3a 20 00 20 87 19   d4
1d11 : a2 18 a0 1d 20 0c e5 a9   d8
1d19 : fa a0 1c 20 1e ab a2 c0   ba
1d21 : a0 c0 88 d0 fd ca d0 f8   c9
1d29 : a5 cb c9 22 f0 07 c9 40   59
1d31 : f0 f6 4c 04 1a a5 fb 48   7f
1d39 : a5 fa 48 a9 00 85 fa a9   0e
1d41 : 20 85 fb a0 00 a9 00 91   a7
1d49 : fa c8 d0 fb e6 fb a5 fb   38
1d51 : c9 28 d0 ef 68 85 fa 68   d0
1d59 : 85 fb 20 04 1a 4c 3f 13   8c
1d61 : 20 87 19 a2 18 a0 1d 20   1b
1d69 : 0c e5 a9 fa a0 1c 20 1e   d9
1d71 : ab a2 c0 a0 c0 88 d0 fd   41
1d79 : ca d0 f8 a5 cb c9 22 f0   14
1d81 : 07 c9 40 f0 f6 4c 04 1a   b1
1d89 : 20 04 1a 20 87 19 a2 18   32
1d91 : a0 1d 20 0c e5 a9 06 a0   4e
1d99 : 1d a2 00 8e 86 02 20 1e   0e
1da1 : ab a9 00 85 c6 20 e4 ff   d3
1da9 : f0 fb 20 e3 1d 20 d8 0b   68
1db1 : ad f1 cf 20 ec 0b 4c 04   af
1db9 : 1a ad 8d 02 29 04 d0 04   4b
1dc1 : ad ea cf 60 a5 cb c9 20   03
1dc9 : d0 08 a2 ff 8e d0 cf 8e   12
1dd1 : ea cf c9 23 d0 ea a2 00   68
1dd9 : 8e d0 cf ca 8e ea cf 4c   35
1de1 : c1 1d 20 81 17 48 ad d0   75
1de9 : cf d0 02 68 60 68 09 80   1c
1df1 : 60 68 68 a0 02 20 9b 0d   5d
1df9 : a0 00 ad ff cf 4c c1 18   9b
1e01 : 68 68 a0 02 20 9b 0d a0   5a
1e09 : 00 ad f0 cf 4c c1 18 68   1a
1e11 : 68 a0 02 20 9b 0d ad e2   ed
1e19 : cf d0 01 60 a9 17 38 ed   ad
1e21 : e2 cf 8d d1 cf ee d1 cf   e4
1e29 : a5 3f 38 e9 01 85 3f a5   3e
1e31 : 40 e9 00 85 40 a0 01 b1   87
1e39 : 3f 88 91 3f c8 c8 cc d1   b2
1e41 : cf d0 f4 88 a9 20 91 3f   27
1e49 : a5 40 18 69 d4 85 40 a0   fd
1e51 : 01 b1 3f 88 91 3f c8 c8   d3
1e59 : cc d1 cf d0 f4 88 ad f1   4a
1e61 : cf 91 3f ce e2 cf a2 02   de
1e69 : 4c 0e 11 68 68 a0 02 20   e2
1e71 : 9b 0d ad e2 cf c9 16 d0   a0
1e79 : 01 60 a9 15 38 ed e2 cf   d5
1e81 : 8d d1 cf ac d1 cf b1 3f   61
1e89 : c8 91 3f 88 88 10 f7 c8   75
1e91 : a9 20 91 3f a5 40 18 69   26
1e99 : d4 85 40 ac d1 cf b1 3f   b6
1ea1 : c8 91 3f 88 88 10 f7 c8   8d
1ea9 : ad f1 cf 91 3f ee e2 cf   0b
1eb1 : a2 02 4c d6 10 a5 fa 48   ed
1eb9 : a5 fb 48 a5 fc 48 a5 fd   c7
1ec1 : 48 a9 00 85 fa 85 fc a9   b2
1ec9 : 20 85 fb a9 28 85 fd a0   c8
1ed1 : 00 b1 fa 48 b1 fc 91 fa   b1
1ed9 : 68 91 fc c8 d0 f3 e6 fb   a2
1ee1 : e6 fd a5 fb c9 28 d0 e7   9f
1ee9 : 68 85 fd 68 85 fc 68 85   8d
1ef1 : fb 68 85 fa 60 93 81 96   b7
1ef9 : 85 ad 8d 85 8e 95 85 b1   78
1f01 : ae a0 9a 85 89 83 88 85   38
```

```
1f09 : 8e 93 81 94 9a b2 ae a0   8f
1f11 : 9a 85 89 83 88 85 8e 86   3d
1f19 : 8f 8c 87 85 b3 ae a0 86   c1
1f21 : 85 8c 84 a0 b2 00 20 87   dc
1f29 : 19 a2 00 bd 38 1a 9d b0   77
1f31 : 06 a9 06 9d b0 da e8 e0   88
1f39 : 12 d0 f0 a2 00 bd 5c 1a   d7
1f41 : 9d d8 06 a9 06 9d d8 da   67
1f49 : e8 e0 12 d0 f0 a2 00 bd   e0
1f51 : 6e 1a 9d 00 07 a9 06 9d   45
1f59 : 00 db e8 e0 12 d0 f0 a2   4e
1f61 : 00 bd 5c 1a 9d 28 07 9d   0d
1f69 : 50 07 9d 78 07 a9 06 9d   c4
1f71 : 28 db 9d 50 db 9d 78 db   3c
1f79 : e8 e0 12 d0 e4 a2 00 bd   4f
1f81 : 80 1a 9d a0 07 a9 06 9d   9b
1f89 : a0 db e8 e0 12 d0 f0 a9   2c
1f91 : a0 8d ea 06 8d 12 07 8d   14
1f99 : 3a 07 8d 62 07 8d 8a 07   1c
1fa1 : 8d b2 07 a9 00 8d ea da   4c
1fa9 : 8d 12 db 8d 3a db 8d 62   65
1fb1 : db 8d 8a db 8d b2 db a2   94
1fb9 : 00 a9 a0 9d c9 07 a9 00   e5
1fc1 : 9d c9 db e8 e0 12 d0 f1   1c
1fc9 : 60 20 27 1f a2 00 bd 00   08
1fd1 : 1f 9d 29 07 bd 0e 1f 9d   ee
1fd9 : 51 07 e8 e0 0e d0 ef bd   a7
1fe1 : 0e 1f 9d 51 07 a2 00 bd   11
1fe9 : f6 1e 9d dc 06 bd 1d 1f   f2
1ff1 : 9d 79 07 e8 e0 09 d0 ef   a3
1ff9 : bd f6 1e 9d dc 06 60 51   8f
2001 : 07 47 41 4d 45 43 48 41   b8
2009 : 52 32 2d 31 30 2b 48 46   f0
2011 : 03 a5 fb 18 69 d4 85 fb   34
2019 : 8a 91 fa a5 fb 38 e9 d4   b2
2021 : 85 fb 60 a5 fa 48 a5 fb   f1
2029 : 48 a9 00 85 c6 ad 40 31   34
2031 : 85 fa ad 41 31 85 fb a0   37
2039 : 00 98 48 a2 07 20 52 31   09
2041 : 20 e4 ff f0 fb aa 68 a8   f9
2049 : 8a c9 14 f0 1f c9 0d f0   31
2051 : 2c 99 42 31 20 81 17 09   7d
2059 : 80 91 fa c0 0f b0 da a2   a0
2061 : 06 20 52 31 a9 a0 c8 91   18
2069 : fa 4c 7a 31 c0 00 f0 c9   b1
2071 : a2 06 20 52 31 a9 a0 91   6f
2079 : fa 88 4c 7a 31 a9 00 85   85
2081 : c6 68 85 fb 68 85 fa 60   bb
2089 : a5 fa 48 a5 fb 48 a5 fc   05
2091 : 48 a5 fd 48 a9 11 85 fa   63
2099 : a9 04 85 fb a9 d0 85 fc   56
20a1 : a9 c2 85 fd a2 00 a0 00   79
20a9 : b1 fa 91 fc c8 c0 17 d0   6c
20b1 : f7 a5 fa 18 69 28 85 fa   20
20b9 : a5 fb 69 00 85 fb a5 fc   7f
20c1 : 18 69 17 85 fc a5 fd 69   cc
20c9 : 00 85 fd e8 e0 0f d0 d6   a0
20d1 : a9 11 85 fa a9 d8 85 fb   33
20d9 : a9 29 85 fc a9 c4 85 fd   ea
20e1 : a2 00 a0 00 b1 fa 91 fc   df
20e9 : c8 c0 17 d0 f7 a5 fa 18   ba
20f1 : 69 28 85 fa a5 fb 69 00   0f
20f9 : 85 fb a5 fc 18 69 17 85   b9
2101 : fc a5 fd 69 00 85 fd e8   72
2109 : e0 0f d0 d6 68 85 fd 68   fb
2111 : 85 fc 68 85 fb 68 85 fa   ee
2119 : 60 ad ea 31 ae ec 31 8e   65
2121 : ea 31 8d ec 31 ad 26 32   22
2129 : ae 28 32 8e 26 32 8d 28   c4
2131 : 32 20 c9 31 ad ea 31 ae   60
2139 : ec 31 8e ea 31 8d ec 31   54
2141 : ad 26 32 ae 28 32 8e 26   fe
```

```
2149 : 32 8d 28 32 60 a9 00 85   f1
2151 : c6 20 ca 1f a2 00 8a 48   a3
2159 : 20 f1 32 20 e4 ff f0 fb   0c
2161 : a8 68 aa 98 c9 0d d0 04   4b
2169 : 8a 18 69 31 c9 11 d0 07   f6
2171 : e8 e0 03 d0 02 a2 00 c9   6d
2179 : 91 d0 07 ca e0 ff d0 02   e3
2181 : a2 02 c9 31 d0 03 4c bd   8f
2189 : 33 c9 32 d0 03 4c 56 36   a0
2191 : c9 33 d0 03 4c 36 33 c9   5f
2199 : 03 f0 0f c9 5f f0 0b c9   4f
21a1 : 20 f0 07 a9 00 85 c6 4c   10
21a9 : 97 32 20 04 1a 4c 3f 13   09
21b1 : e0 00 d0 05 a9 07 b8 50   bc
21b9 : 02 a9 06 a0 00 99 29 db   4f
21c1 : c8 c0 10 d0 f8 e0 01 d0   44
21c9 : 05 a9 07 b8 50 02 a9 06   44
21d1 : a0 00 99 51 db c8 c0 10   29
21d9 : d0 f8 e0 02 d0 05 a9 07   88
21e1 : b8 50 02 a9 06 a0 00 99   10
21e9 : 79 db c8 c0 10 d0 f8 60   c6
21f1 : 8e 81 8d 85 ba a9 00 85   58
21f9 : c6 20 2a 1f a2 00 bd 1f   9d
2201 : 1f 9d dc 06 e8 e0 07 d0   3a
2209 : f5 a2 00 bd 31 33 9d 29   7d
2211 : 07 e8 e0 05 d0 f5 a9 07   d6
2219 : 8d 41 31 a9 51 8d 40 31   ad
2221 : 20 64 31 c0 01 d0 0d ad   fe
2229 : 42 31 c9 5f d0 06 20 04   28
2231 : 1a 4c 3f 13 98 a2 42 a0   8c
2239 : 31 20 bd ff a9 01 a2 08   27
2241 : a0 01 20 ba ff 20 c9 31   4c
2249 : ad 11 d0 29 ef 8d 11 d0   29
2251 : a5 fa 48 a5 fb 48 a9 d0   84
2259 : 85 fa a9 c2 85 fb a9 fa   f3
2261 : a2 82 a0 c5 20 e9 35 68   1c
2269 : 85 fb 68 85 fa 20 04 1a   ac
2271 : 20 3f 13 ad 11 d0 09 10   87
2279 : 8d 11 d0 60 a9 00 85 c6   0d
2281 : 20 2a 1f a2 00 bd 02 1f   07
2289 : 9d da 06 e8 e0 0c d0 f5   d0
2291 : a2 00 bd 31 33 9d 29 07   9c
2299 : e8 e0 05 d0 f5 a9 07 8d   31
22a1 : 41 31 a9 51 8d 40 31 20   ef
22a9 : 64 31 c0 01 d0 0d ad 42   a7
22b1 : 31 c9 5f d0 06 20 04 1a   5e
22b9 : 4c 3f 13 98 a2 42 a0 31   9e
22c1 : 20 bd ff a9 01 a2 08 a0   7b
22c9 : 01 20 ba ff 20 c9 31 ad   f9
22d1 : 11 d0 29 ef 8d 11 d0 a5   83
22d9 : fa 48 a5 fb 48 a9 00 85   bd
22e1 : fa a9 20 85 fb a9 fa a2   a7
22e9 : 00 a0 28 20 e9 35 68 85   3c
22f1 : fb 68 85 fa 20 04 1a 20   ac
22f9 : 3f 13 ad 11 d0 09 10 8d   00
2301 : 11 d0 60 8c 8f 81 84 ad   96
2309 : 8d 85 8e 95 85 a9 34 8d   41
2311 : ea 1f 8d fb 1f a9 44 8d   d9
2319 : e9 1f 8d fa 1f 20 ca 1f   b1
2321 : a9 1e 8d ea 1f 8d fb 1f   27
2329 : a9 f6 8d e9 1f 8d fa 1f   76
2331 : a9 00 85 c6 a2 00 8a 48   f9
2339 : 20 f1 32 20 e4 ff f0 fb   ec
2341 : a8 68 aa 98 c9 0d d0 04   2b
2349 : 8a 18 69 31 c9 11 d0 07   d6
2351 : e8 e0 03 d0 02 a2 00 c9   4d
2359 : 91 d0 07 ca e0 ff d0 02   c3
2361 : a2 02 c9 31 d0 03 4c 55   9e
2369 : 35 c9 32 d0 03 4c b5 37   01
2371 : c9 33 d0 03 4c d1 34 c9   20
2379 : 03 f0 0f c9 5f f0 0b c9   2f
2381 : 20 f0 07 a9 00 85 c6 4c   f0
```

```
2389 : 77 34 20 04 1a 4c 3f 13   ca
2391 : a9 00 85 c6 20 2a 1f a2   8a
2399 : 00 bd 1f 1f 9d dc 06 e8   ce
23a1 : e0 07 d0 f5 a2 00 bd 31   7b
23a9 : 33 9d 29 07 e8 e0 05 d0   21
23b1 : f5 a9 07 8d 41 31 a9 51   d5
23b9 : 8d 40 31 20 64 31 c0 01   8c
23c1 : d0 0d ad 42 31 c9 5f d0   4c
23c9 : 06 20 04 1a 4c 3f 13 98   60
23d1 : a2 42 a0 31 20 bd ff a9   26
23d9 : 01 a2 08 a0 00 20 ba ff   2d
23e1 : 20 c9 31 ad 11 d0 29 ef   04
23e9 : 8d 11 d0 a5 fa 48 a5 fb   68
23f1 : 48 a9 00 a2 d0 a0 c2 20   c0
23f9 : d6 35 68 85 fb 68 85 fa   44
2401 : 20 04 1a 20 3f 13 20 5a   70
2409 : 32 ad 11 d0 09 10 8d 11   da
2411 : d0 4c fb 0a a9 00 85 c6   86
2419 : 20 2a 1f a2 00 bd 02 1f   9f
2421 : 9d da 06 e8 e0 0c d0 f5   68
2429 : a2 00 bd 31 33 9d 29 07   34
2431 : e8 e0 05 d0 f5 a9 07 8d   c9
2439 : 41 31 a9 51 8d 40 31 20   87
2441 : 64 31 c0 01 d0 0d ad 42   3f
2449 : 31 c9 5f d0 06 20 04 1a   f6
2451 : 4c 3f 13 98 a2 42 a0 31   36
2459 : 20 bd ff a9 01 a2 08 a0   13
2461 : 00 20 ba ff ea ea ea ad   2d
2469 : 11 d0 29 ef 8d 11 d0 a5   1b
2471 : fa 48 a5 fb 48 a2 00 a0   53
2479 : 20 a9 00 20 d6 35 68 85   36
2481 : fb 68 85 fa 20 04 1a 20   3c
2489 : 3f 13 ad 11 d0 09 10 8d   90
2491 : 11 d0 4c fb 0a 48 8a 48   3a
2499 : 98 48 20 8a 0b 68 a8 68   16
24a1 : aa 68 20 d5 ff 4c ad 37   ca
24a9 : 48 8a 48 98 48 20 8a 0b   21
24b1 : 68 a8 68 aa 68 20 d8 ff   c8
24b9 : 4c ad 37 85 3f a9 00 85   a7
24c1 : 40 06 3f 26 40 06 3f 26   16
24c9 : 40 06 3f 26 40 a5 40 18   03
24d1 : 69 20 85 40 60 a5 3f 48   74
24d9 : a5 40 48 ad fd cf 20 fc   3f
24e1 : 35 a0 00 b1 3f 91 fa c8   9b
24e9 : c0 08 d0 f7 68 85 40 68   65
24f1 : 85 3f 4c 3f 13 a5 3f 48   fd
24f9 : a5 40 48 ad ff cf 20 fc   7f
2501 : 35 a0 00 b1 3f 91 fa c8   bb
2509 : c0 08 d0 f7 68 85 40 68   85
2511 : 85 3f 4c 3f 13 a9 00 85   ba
2519 : c6 20 2a 1f a2 00 bd 10   9f
2521 : 1f 9d da 06 e8 e0 0d d0   f2
2529 : f5 a2 00 bd 31 33 9d 29   9d
2531 : 07 e8 e0 05 d0 f5 a9 07   f6
2539 : 8d 41 31 a9 51 8d 40 31   cd
2541 : 20 64 31 c0 01 d0 0d ad   1e
2549 : 42 31 c9 5f d0 06 20 04   48
2551 : 1a 4c 3f 13 98 a2 42 a0   ac
2559 : 31 20 bd ff a9 01 a2 08   47
2561 : a0 01 20 ba ff 20 04 1a   26
2569 : 20 33 13 20 84 11 20 e4   07
2571 : ff c9 0d f0 10 c9 03 f0   f3
2579 : 0b c9 5f f0 07 c9 20 f0   80
2581 : 03 4c a9 36 60 a9 00 85   3a
2589 : c6 ad 00 d0 8d 06 d0 ad   e8
2591 : 01 d0 8d 07 d0 a9 30 8d   75
2599 : fb 07 a9 0f 8d 15 d0 20   69
25a1 : 33 13 20 84 11 20 e4 ff   9c
```

Listing 1. »Character-Editor« (Fortsetzung)

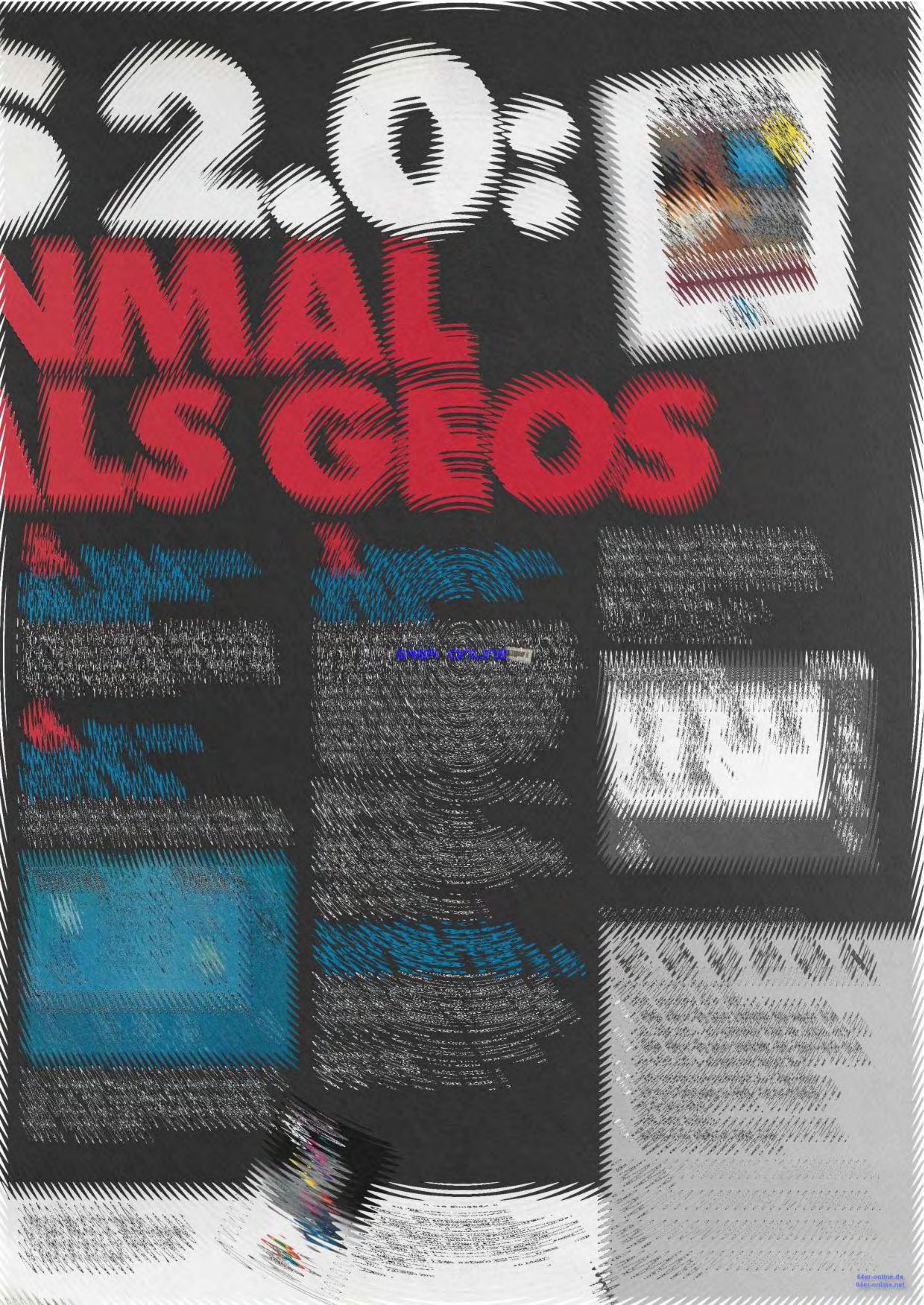
```
25a9 : c9 0d f0 1d c9 20 f0 0b  50
25b1 : c9 03 f0 07 c9 5f f0 03  7a
25b9 : 4c e0 36 a9 07 8d 15 d0  0b
25c1 : a9 00 8d 06 d0 8d 07 d0  c6
25c9 : 60 a5 3f 48 a5 40 48 a0  93
25d1 : 04 20 9b 0d 20 fc 35 a5  78
25d9 : 3f 8d d2 cf a5 40 8d d3  c8
25e1 : cf a0 01 20 9b 0d 20 fc  e1
25e9 : 35 a5 3f 8d d4 cf a5 40  55
25f1 : 8d d5 cf ad d5 cf cd d3  cd
25f9 : cf 90 30 d0 08 ad d4 cf  17
2601 : cd d2 cf 90 26 ad d2 cf  f8
2609 : 85 3f ad d3 cf 85 40 ad  99
2611 : d4 cf 18 69 08 8d d4 cf  e0
2619 : ad d5 cf 69 00 8d d5 cf  35
2621 : a9 3f ae d4 cf ac d5 cf  09
2629 : 4c 8f 37 ad d4 cf 85 3f  21
2631 : ad d5 cf 85 40 ad d2 cf  ca
2639 : 18 69 08 8d d2 cf ad d3  c4
2641 : cf 69 00 8d d3 cf a9 3f  57
2649 : ae d2 cf ac d3 cf ad 11  7e
2651 : d0 29 ef 8d 11 d0 a9 3f  20
2659 : 20 e9 35 ad 11 d0 09 10  4d
2661 : 8d 11 d0 68 85 40 68 85  bf
2669 : 3f 4c fc 36 20 a3 fd 4c  84
2671 : 62 3b ea ea a9 00 85 c6  c7
2679 : 20 2a 1f a2 00 bd 10 1f  37
2681 : 9d da 06 e8 e0 0d d0 f5  d0
2689 : a2 00 bd 31 33 9d 29 07  94
2691 : e8 e0 05 d0 f5 a9 07 8d  29
2699 : 41 31 a9 51 8d 40 31 20  e7
26a1 : 64 31 c0 01 d0 0d ad 42  9f
26a9 : 31 c9 5f d0 06 20 04 1a  56
26b1 : 4c 3f 13 98 a2 42 a0 31  96
26b9 : 20 bd ff a9 01 a2 08 a0  73
26c1 : 00 20 ba ff 20 04 1a 20  4b
26c9 : 33 13 20 84 11 20 e4 ff  c4
26d1 : c9 0d f0 10 c9 03 f0 0b  ed
26d9 : c9 5f f0 07 c9 20 f0 03  d6
26e1 : 4c 08 38 60 a5 3f 48 a5  0c
26e9 : 40 48 a0 01 20 9b 0d 20  e9
26f1 : fc 35 ad 11 d0 29 ef 8d  46
26f9 : 11 d0 a9 00 a6 3f a4 40  54
2701 : 20 d6 35 ad 11 d0 09 10  6b
2709 : 8d 11 d0 68 85 40 68 85  67
2711 : 3f 60 84 89 93 8b ad 8d  3a
2719 : 85 8e 95 85 b1 ae a0 84  17
2721 : 89 92 85 83 94 8f 92 99  08
2729 : b2 ae a0 84 89 93 8b 82  53
2731 : 85 86 85 88 8c a9 01 85  91
2739 : c7 a2 08 a0 0f 20 ba ff  44
2741 : a9 00 20 bd ff 20 c0 ff  ae
2749 : a2 01 20 c6 ff 20 cf ff  8d
2751 : 20 d2 ff 24 90 50 f6 20  06
2759 : cc ff a9 00 85 c7 a9 01  ce
2761 : 4c c3 ff ea ea ea 20 27  c1
2769 : 1f a2 00 bd 53 38 9d dc  b8
2771 : 06 e8 e0 0a d0 f5 a2 00  ac
2779 : bd 5d 38 9d 29 07 e8 e0  d7
2781 : 0c d0 f5 a2 00 bd 69 38  cb
2789 : 9d 51 07 e8 e0 0d d0 f5  53
2791 : a2 16 a0 09 20 0c e5 a9  d5
2799 : 06 8d 86 02 20 20 3b a2  7d
27a1 : 00 8a 48 20 f1 32 20 e4  f7
27a9 : ff f0 fb a8 68 aa 98 c9  06
27b1 : 0d d0 04 8a 18 69 31 c9  9e
27b9 : 11 d0 07 e8 e0 03 d0 02  7f
27c1 : a2 00 c9 91 d0 07 ca e0  3a
27c9 : ff d0 02 a2 02 c9 31 d0  da
27d1 : 03 4c be 39 c9 32 d0 03  49
27d9 : 4c 38 43 c9 33 d0 03 4c  aa
27e1 : 2e 3b c9 03 f0 0b c9 5f  cd
27e9 : f0 07 c9 20 f0 03 4c 06  38
27f1 : 3b 4c 0d 3b a5 fa 48 a5  9c
27f9 : fb 48 a5 fc 48 a5 fd 48  5b
2801 : a9 00 85 fa 85 fc a9 04  5a
2809 : 85 fb a9 c6 85 fd a0 00  9a
2811 : b1 fa 91 fc c8 d0 f9 e6  0c
2819 : fb e6 fd a5 fb c9 08 d0  8b
2821 : ef a9 d8 85 fb a9 ca 85  0f
2829 : fd a0 00 b1 fa 91 fc c8  6e
2831 : d0 f9 e6 fb e6 fd a5 fb  24
2839 : c9 dc d0 ed 68 85 fd 68  de
2841 : 85 fc 68 85 fb 68 85 fa  1e
2849 : 60 ad 52 39 ae 54 39 8e  cb
2851 : 52 39 8d 54 39 ad 6d 39  57
2859 : ae 6f 39 8e 6d 39 8d 6f  95
2861 : 39 20 35 39 ad 52 39 ae  cf
2869 : 54 39 8e 52 39 8d 54 39  0b
2871 : ad 6d 39 ae 6f 39 8e 6d  cf
2879 : 39 8d 6f 39 60 20 04 1a  c7
2881 : ad 15 d0 48 a9 00 8d 15  f1
2889 : d0 20 35 39 20 8a 0b a9  b4
2891 : 07 8d 86 02 20 44 e5 20  3d
2899 : d4 09 a9 00 85 c6 20 e4  35
28a1 : ff f0 fb 20 fb 0a 20 8a  c1
28a9 : 39 68 8d 15 d0 60 f0 c0  72
28b1 : c0 c0 c0 c0 c0 c0 c0 c0  b0
28b9 : c0 c0 c0 c0 c0 c0 c0 c0  b8
28c1 : c0 c0 c0 c0 c0 c0 c0 c0  c0
28c9 : c0 c0 c0 c0 c0 ee dd a0  6e
28d1 : a0 a0 a0 a0 a0 a0 a0 a0  d0
28d9 : a0 a0 a0 a0 a0 a0 a0 a0  d8
28e1 : a0 a0 a0 a0 a0 a0 a0 a0  e0
28e9 : a0 a0 a0 a0 a0 dd eb c0  40
28f1 : c0 c0 c0 c0 c0 c0 c0 c0  f0
28f9 : c0 c0 c0 c0 c0 c0 c0 c0  f8
2901 : c0 c0 c0 c0 c0 c0 c0 c0  00
2909 : c0 c0 c0 c0 c0 f3 ed c0  56
2911 : c0 c0 c0 c0 c0 c0 c0 c0  10
2919 : c0 c0 c0 c0 c0 c0 c0 c0  18
2921 : c0 c0 c0 c0 c0 c0 c0 c0  20
2929 : c0 c0 c0 c0 c0 fd 20 87  1d
2931 : 19 a2 00 bd ef 39 9d ac  ec
2939 : 06 a9 06 9d ac da e8 e0  50
2941 : 20 d0 f0 a2 00 bd 0f 3a  f8
2949 : 9d d4 06 a9 06 9d d4 da  5d
2951 : e8 e0 20 d0 f0 a9 a0 9d  e6
2959 : d4 06 a9 00 9d d4 da a2  cc
2961 : 00 bd 2f 3a 9d fc 06 a9  80
2969 : 06 9d fc da e8 e0 20 d0  90
2971 : f0 a9 a0 9d fc 06 a9 00  b8
2979 : 9d fc da a2 00 bd 0f 3a  3e
2981 : 9d 24 07 a9 06 9d 24 db  bd
2989 : e8 e0 20 d0 f0 a9 a0 9d  1e
2991 : 24 07 a9 00 9d 24 db a2  53
2999 : 00 bd 4f 3a 9d 4c 07 a9  3f
29a1 : 06 9d 4c db e8 e0 20 d0  bc
29a9 : f0 a9 a0 9d 4c 07 a9 00  ed
29b1 : 9d 4c db a2 00 a9 a0 9d  cb
29b9 : 75 07 a9 00 9d 75 db e8  e3
29c1 : e0 20 d0 f1 60 a9 00 85  82
29c9 : c6 4c e2 38 20 04 1a 4c  98
29d1 : 3f 13 b3 ae a0 84 89 93  d8
29d9 : 8b 93 94 81 94 95 93 a2  0d
29e1 : 00 bd 13 3b 9d 79 07 e8  80
29e9 : e0 0d d0 f5 60 20 04 1a  8e
29f1 : 20 87 19 20 7d 3b a2 00  5b
29f9 : bd 13 3b 9d dd 06 e8 e0  36
2a01 : 10 d0 f5 a2 14 a0 05 20  e3
2a09 : 0c e5 a9 06 8d 86 02 20  88
2a11 : 8b 3b a9 00 85 c6 20 e4  7d
2a19 : ff f0 fb 20 04 1a 4c c9  69
2a21 : 3c 20 b2 43 20 99 3b 20  7e
2a29 : be 43 ea a9 81 8d 0e dc  ef
2a31 : a9 01 8d 1a d0 8d 19 d0  81
2a39 : ad 0d dc 60 20 8a 0b ad  8e
2a41 : 11 d0 29 ef 8d 11 d0 4c  40
2a49 : bb 3c 20 76 38 ad 11 d0  d0
2a51 : 09 10 8d 11 d0 4c 62 3b  57
2a59 : a2 10 a0 10 88 d0 fd ca  ca
2a61 : d0 f8 60 11 91 91 91 91  f7
2a69 : 91 91 91 11 03 45 20 57  d3
2a71 : 49 45 20 57 41 45 52 45  62
2a79 : 20 45 53 20 4d 49 54 20  c5
2a81 : 45 41 07 07 a5 fb 18 69  77
2a89 : d4 85 fb 8a 91 fa a5 fb  ef
2a91 : 38 e9 d4 85 fb 60 a5 fa  f3
2a99 : 48 a5 fb 48 a9 00 85 c6  fa
2aa1 : ad 3c 3c 85 fa ad 3d 3c  b7
2aa9 : 85 fb a0 00 98 48 a2 07  b8
2ab1 : 20 c5 3b 20 e4 ff f0 fb  91
2ab9 : aa 68 a8 8a c9 14 f0 1f  52
2ac1 : c9 0d f0 2c 99 a4 3b 20  be
2ac9 : 81 17 09 80 91 fa c0 1d  56
2ad1 : b0 da a2 06 20 c5 3b a9  c8
2ad9 : a0 c8 91 fa 4c ed 3b c0  44
2ae1 : 00 f0 c9 a2 06 20 c5 3b  0f
2ae9 : a9 a0 91 fa 88 4c ed 3b  bf
2af1 : a9 00 85 c6 68 85 fb 68  48
2af9 : 85 fa 60 25 07 20 04 1a  6e
2b01 : 20 87 19 20 6f 3a a2 00  83
2b09 : bd 6c 38 9d df 06 e8 e0  52
2b11 : 0a d0 f5 a9 25 8d 3c 3c  5e
2b19 : a9 07 8d 3d 3c 20 d7 3b  eb
2b21 : c0 00 d0 03 4c 04 1a c0  45
2b29 : 01 d0 07 ad a4 3b c9 5f  14
2b31 : f0 f2 20 8a 0b ad 11 d0  f8
2b39 : 29 ef 8d 11 d0 8c c4 3b  da
2b41 : a9 01 a2 08 a0 0f 20 ba  8d
2b49 : ff a9 00 20 bd ff 20 c0  ff
2b51 : ff a2 01 20 c9 ff a2 00  0d
2b59 : bd a4 3b 20 d2 ff e8 ec  e6
2b61 : c4 3b d0 f4 20 cc ff a9  51
2b69 : 01 20 c3 ff 20 04 1a ad  51
2b71 : 11 d0 09 10 8d 11 d0 4c  6c
2b79 : ad 37 ad 15 d0 8d c3 3b  cf
2b81 : a9 00 8d 15 d0 4c 6f 3a  d2
2b89 : ad c3 3b 8d 15 d0 4c ad  fd
2b91 : 37 95 93 85 92 ad 8d 85  00
2b99 : 8e 95 85 b1 ae a0 83 8f  a6
2ba1 : 90 99 a0 9a 85 89 83 88  3d
2ba9 : 85 8e b2 ae a0 83 8c 85  5b
2bb1 : 81 92 a0 9a 85 89 83 88  ba
2bb9 : 85 8e b3 ae a0 89 8e 96  06
2bc1 : 94 a0 9a 85 89 83 88 85  de
2bc9 : 8e 20 87 19 20 27 1f a2  69
2bd1 : 00 bd d2 3c 9d dc 06 e8  97
2bd9 : e0 0a d0 f5 a2 00 bd dc  8c
2be1 : 3c 9d 29 07 e8 e0 0f d0  8a
2be9 : f5 a2 00 bd eb 3c 9d 51  a1
2bf1 : 07 e8 e0 10 d0 f5 a2 00  ee
2bf9 : bd fb 3c 9d 79 07 e8 e0  ac
2c01 : 0f d0 f5 a9 00 85 c6 a2  b7
2c09 : 00 8a 48 20 f1 32 20 e4  5f
2c11 : ff f0 fb a8 68 aa 98 c9  6e
2c19 : 0d d0 04 8a 18 69 31 c9  06
2c21 : 11 d0 07 e8 e0 03 d0 02  e7
2c29 : a2 00 c9 91 d0 07 ca e0  a2
2c31 : ff d0 02 a2 02 c9 31 d0  42
2c39 : 03 4c 6d 3e c9 32 d0 03  fd
2c41 : 4c 5b 3f c9 33 d0 03 4c  a2
2c49 : c5 3f c9 03 f0 0f c9 5f  ee
2c51 : f0 0b c9 20 f0 07 a9 00  2b
2c59 : 85 c6 4c 4a 3d 20 04 1a  b7
2c61 : 4c 3f 13 c0 01 f0 0f c0  7f
```

```
2c69 : 39 f0 0b c0 3f f0 07 c0   0e
2c71 : 3c f0 03 4c 9c 1c 4c 96   79
2c79 : 1c 20 33 13 20 84 11 20   7f
2c81 : e4 ff c9 0d f0 0f c9 03   2e
2c89 : f0 0d c9 5f f0 09 c9 20   1d
2c91 : f0 05 4c ba 3d 18 60 38   f5
2c99 : 60 a5 3f 48 a5 40 48 a0   63
2ca1 : 04 20 9b 0d 20 fc 35 a5   48
2ca9 : 3f 8d d2 cf a5 40 8d d3   98
2cb1 : cf a0 01 20 9b 0d 20 fc   b1
2cb9 : 35 a5 3f 8d d4 cf a5 40   25
2cc1 : 8d d5 cf ad d5 cf cd d3   9d
2cc9 : cf 90 30 d0 08 ad d4 cf   e7
2cd1 : cd d2 cf 90 26 ad d2 cf   c8
2cd9 : 85 3f ad d3 cf 85 40 ad   69
2ce1 : d4 cf 18 69 08 8d d4 cf   b0
2ce9 : ad d5 cf 69 00 8d d5 cf   05
2cf1 : a9 3f ae d4 cf ac d5 cf   d9
2cf9 : 4c 5f 3e ad d4 cf 85 3f   9a
2d01 : ad d5 cf 85 40 ad d2 cf   9a
2d09 : 18 69 08 8d d2 cf ad d3   94
2d11 : cf 69 00 8d d3 cf a9 3f   27
2d19 : ae d2 cf ac d3 cf 60 a2   3c
2d21 : 00 a9 20 9d e3 07 e8 e0   8d
2d29 : 05 d0 f8 60 20 04 1a a2   b0
2d31 : 00 bd df 3c 9d e4 07 e8   7e
2d39 : e0 04 d0 f5 20 ba 3d 90   fc
2d41 : 03 4c 60 3e a9 00 85 c6   88
2d49 : ad 00 d0 8d 06 d0 ad 01   7c
2d51 : d0 8d 07 d0 a9 30 8d fb   0e
2d59 : 07 a9 0f 8d 15 d0 20 ba   78
2d61 : 3d 90 06 20 60 3e 4c fc   8f
2d69 : 36 20 5f 3e a9 00 85 c6   8d
2d71 : ad 00 d0 8d 08 d0 ad 01   c4
2d79 : d0 8d 09 d0 a9 30 8d fc   b8
2d81 : 07 a9 1f 8d 15 d0 20 ba   a4
2d89 : 3d 90 12 a9 00 8d 08 d0   f6
2d91 : 8d 09 d0 20 60 3e 4c fc   fe
2d99 : 36 00 50 b0 50 a5 3f 48   b9
2da1 : a5 40 48 a5 fa 48 a5 fb   ae
2da9 : 48 a9 05 8d f3 3d 20 e0   24
2db1 : 3d a5 3f 8d da 3e a5 40   f9
2db9 : 8d db 3e 8e dc 3e 8c dd   43
2dc1 : 3e a9 01 8d f3 3d a0 01   73
2dc9 : 20 9b 0d 20 fc 35 20 55   a3
2dd1 : 43 85 fa ad db 3e 85 fb   09
2dd9 : a0 00 b1 fa 91 3f a5 3f   6d
2de1 : 18 69 01 85 3f 90 02 e6   ed
2de9 : 40 a5 fa 18 69 01 85 fa   68
2df1 : 90 02 e6 fb a5 40 c9 28   8f
2df9 : f0 0e a5 fb cd dd 3e d0   3f
2e01 : d7 a5 fa cd dc 3e d0 d0   c8
2e09 : 68 85 fb 68 85 fa 68 85   1d
2e11 : 40 68 85 3f 20 3f 13 4c   af
2e19 : cc 3e 20 04 1a a2 00 bd   bf
2e21 : ee 3c 9d e3 07 e8 e0 05   56
2e29 : d0 f5 20 ba 3d 90 03 4c   50
2e31 : 60 3e a9 00 85 c6 ad 00   60
2e39 : d0 8d 06 d0 ad 01 d0 8d   ad
2e41 : 07 d0 a9 30 8d fb 07 a9   49
2e49 : 0f 8d 15 d0 20 ba 3d 90   6c
2e51 : 03 4c a4 3e a5 3f 48 a5   2c
2e59 : 40 48 20 e0 3d 8e da 3e   12
2e61 : 8c db 3e a9 00 a8 91 3f   aa
2e69 : a5 3f 18 69 01 85 3f 90   3b
2e71 : 02 e6 40 a5 40 cd db 3e   09
2e79 : d0 e9 a5 3f cd da 3e d0   dd
2e81 : e2 4c 4f 3f 20 04 1a a2   15
2e89 : 00 bd fe 3c 9d e4 07 e8   9e
2e91 : e0 04 d0 f5 20 ba 3d 90   54
2e99 : 03 4c 60 3e a9 00 85 c6   e0
2ea1 : ad 00 d0 8d 06 d0 ad 01   d4
2ea9 : d0 8d 07 d0 a9 30 8d fb   66
2eb1 : 07 a9 0f 8d 15 d0 20 ba   d0
2eb9 : 3d 90 03 4c a4 3e a5 3f   da
2ec1 : 48 a5 40 48 20 e0 3d 8e   10
2ec9 : da 3e 8c db 3e a0 00 b1   ad
2ed1 : 3f 49 ff 91 3f a5 3f 18   35
2ed9 : 69 01 85 3f 90 02 e6 40   41
2ee1 : a5 40 cd db 3e d0 e8 a5   ee
2ee9 : 3f cd da 3e d0 e1 4c 4f   79
2ef1 : 3f 20 87 19 a2 18 a0 1d   ed
2ef9 : 20 0c e5 a9 5c a0 40 20   da
2f01 : 1e ab a9 00 85 c6 20 e4   38
2f09 : ff f0 fb c9 4a f0 03 4c   89
2f11 : 04 1a 20 d5 0e 20 04 1a   0b
2f19 : 4c 3f 13 57 49 52 4b 4c   a2
2f21 : 49 43 48 3f 00 20 87 19   57
2f29 : a2 18 a0 1d 20 0c e5 a9   f0
2f31 : 5c a0 40 20 1e ab a9 00   d7
2f39 : 85 c6 20 e4 ff f0 fb c9   d1
2f41 : 4a f0 03 4c 04 1a a9 28   56
2f49 : 8d 54 0a a9 d8 8d 4c 0a   f7
2f51 : a9 e0 8d 50 0a 20 42 0a   96
2f59 : a9 d0 8d 4c 0a a9 d8 8d   c4
2f61 : 50 0a 20 04 1a 4c 3f 13   66
2f69 : 20 40 17 ad f4 cf d0 01   38
2f71 : 60 ad fb cf f0 0a a9 00   a7
2f79 : 8d fb cf a9 31 4c c8 40   46
2f81 : a9 ff 8d fb cf a9 30 8d   33
2f89 : fa 07 a5 fa 48 a5 fb 48   02
2f91 : a9 00 85 fa a9 d8 85 fb   6a
2f99 : a2 00 a0 00 b1 fa 49 08   8c
2fa1 : 91 fa c8 c0 10 d0 f5 a5   a4
2fa9 : fa 18 69 28 85 fa 90 02   85
2fb1 : e6 fb e8 e0 10 d0 e3 68   d3
2fb9 : 85 fb 68 85 fa 20 4d 10   0d
2fc1 : 4c 3f 13 8d 86 02 [illegible] 00   42
2fc9 : 85 c7 60 a9 07 8d 86 02   7a
2fd1 : 20 44 e5 a2 00 8a 9d d0   4d
2fd9 : cf e8 e0 41 d0 f8 a9 07   06
2fe1 : 8d f1 cf 8d f5 cf 8d fe   1e
2fe9 : cf a9 2e 8d fd cf a9 01   d1
2ff1 : 8d fc cf 20 ae 0a a9 2e   32
2ff9 : 20 d8 0b a9 17 8d 00 d0   fd
3001 : a9 31 8d 01 d0 a9 9f 8d   ba
3009 : 02 d0 a9 31 8d 03 d0 a9   8b
3011 : 17 8d 04 d0 a9 b9 8d 05   b3
3019 : d0 a9 30 8d f8 07 8d f9   6d
3021 : 07 8d fa 07 a9 07 8d 15   c2
3029 : d0 a9 00 8d 10 d0 a9 00   ae
3031 : 8d c7 42 ea 20 8a 0b 20   52
3039 : fb 0a 20 3f 13 20 4d 10   b1
3041 : 20 fe 11 ad fc cf c9 01   52
3049 : f0 0b c9 02 f0 0d c9 03   16
3051 : f0 0f 4c a8 41 20 e0 11   ac
3059 : 4c a8 41 20 9d 18 4c a8   6b
3061 : 41 20 7e 15 4c a8 41 4c   9c
3069 : c8 42 ad 8d 02 29 01 f0   bf
3071 : 03 4c 6e 42 a5 cb c9 2f   bc
3079 : d0 03 4c e2 18 c9 2c d0   5c
3081 : 03 4c 07 19 c9 1f d0 03   6e
3089 : 4c b4 10 c9 28 d0 03 4c   1a
3091 : 14 19 c9 2b d0 03 4c 1a   94
3099 : 19 c9 15 d0 06 20 17 17   e2
30a1 : 4c 3f 13 c9 21 d0 06 20   7c
30a9 : a1 16 4c 3f 13 c9 31 d0   36
30b1 : 06 20 b2 16 4c 3f 13 c9   d5
30b9 : 36 d0 06 20 d3 16 4c 3f   7b
30c1 : 13 c9 33 d0 03 4c 40 17   61
30c9 : c9 24 d0 03 4c 2c 19 c9   57
30d1 : 0d d0 03 4c 8e 32 c9 1d   6d
30d9 : d0 03 4c a9 40 c9 14 d0   b7
30e1 : 03 4c 5b 1c c9 2d d0 06   ba
30e9 : 20 91 15 4c 3f 13 c9 32   b9
30f1 : d0 06 20 1a 16 4c 3f 13   f7
30f9 : c9 09 d0 06 20 b6 1e 4c   05
3101 : 3f 13 c9 2a d0 03 4c 4e   74
3109 : 34 c9 12 d0 03 4c a7 38   62
3111 : c9 1e d0 03 4c 0a 3d c9   1b
3119 : 1a d0 03 4c 32 40 c9 25   7c
3121 : d0 03 4c 66 40 c9 0a d0   6f
3129 : 03 4c 0e 43 60 c0 33 d0   b9
3131 : 06 20 93 16 4c 3f 13 c0   7c
3139 : 31 d0 06 20 49 16 4c 3f   4d
3141 : 13 c0 36 d0 06 20 54 16   3b
3149 : 4c 3f 13 c0 2c d0 03 4c   00
3151 : 16 36 c0 2f d0 03 4c 36   5b
3159 : 36 c0 12 d0 03 4c c1 39   9a
3161 : c0 0d d0 03 20 31 3b 60   76
3169 : ad f4 cf d0 04 ad f5 cf   c3
3171 : 60 ad f5 cf 29 07 60 ad   c7
3179 : f4 cf d0 04 ad fe cf 60   dc
3181 : ad fe cf 29 07 60 00 ad   95
3189 : c7 42 d0 03 4c 81 41 ad   37
3191 : f0 cf 20 f7 42 8d 79 07   f4
3199 : 8e 78 07 ad ff cf 20 f7   ca
31a1 : 42 8d 83 07 8e 82 07 ad   e0
31a9 : fd cf 20 f7 42 8d 8d 07   6a
31b1 : 8e 8c 07 4c 81 41 48 4a   a9
31b9 : 4a 4a 4a 20 01 43 aa 68   65
31c1 : 29 0f c9 0a 90 04 38 e9   03
31c9 : 09 60 69 30 60 a2 b0 a0   82
31d1 : b0 88 d0 fd ca d0 f8 ad   2b
31d9 : c7 42 49 ff 8d c7 42 ad   8f
31e1 : c7 42 d0 12 a9 20 a2 00   66
31e9 : 9d 78 07 9d 82 07 9d 8c   28
31f1 : 07 e8 e0 02 d0 f2 60 ad   66
31f9 : 15 d0 48 ad 15 d0 29 fb   b3
3201 : 8d 15 d0 20 3e 3c 68 8d   d3
3209 : 15 d0 60 8d f1 cf 20 4d   09
3211 : 10 4c 3f 13 ad da 3e 85   2f
3219 : fa ad db 3e 85 fb a5 3f   f6
3221 : 48 a5 40 48 a9 00 85 3f   84
3229 : a9 50 85 40 a0 00 b1 fa   2a
3231 : 91 3f a5 3f 18 69 01 85   8f
3239 : 3f 90 02 e6 40 a5 fa 18   6b
3241 : 69 01 85 fa 90 02 e6 fb   98
3249 : a5 fb cd dd 3e d0 dd a5   48
3251 : fa cd dc 3e d0 d6 a5 3f   0a
3259 : 8d dc 3e a5 40 8d dd 3e   fd
3261 : 68 85 40 68 85 3f a9 50   43
3269 : 8d db 3e a9 00 8d da 3e   fd
3271 : 60 20 8a 0b ad 11 d0 09   9e
3279 : 10 8d 11 d0 60 20 fb 0a   b9
3281 : ad 12 d0 c9 50 d0 f9 ad   73
3289 : f2 cf 09 f0 cd 21 d0 f0   ce
3291 : 03 4c be 43 ad 12 d0 c9   15
3299 : c0 d0 f9 ad 21 d0 c9 fc   af
32a1 : f0 03 4c be 43 60 a9 00   dc
32a9 : 85 9d 4c bb 0b ad fb cf   35
32b1 : d0 06 ad f1 cf 09 08 60   54
32b9 : ad f1 cf 29 f7 60 a2 00   85
32c1 : bd cf 32 9d 34 03 e8 e0   67
32c9 : 38 d0 f5 4c 34 03 a9 00   72
32d1 : 85 5f a9 20 85 60 a9 bf   f6
32d9 : 85 5a a9 32 85 5b a9 ff   16
32e1 : 85 58 a9 43 85 59 20 bf   88
32e9 : a3 20 cd 0e 20 d5 0e a9   0e
32f1 : 0c 8d c1 08 a9 41 8d c2   96
32f9 : 08 a9 e7 8d 26 08 a9 43   51
3301 : 8d 27 08 4c 0c 41 98 67   a9
```

Listing 1. »Character-Editor« (Schluß)



GEOS
SIEBE
BESSER A
64er online



GIGA PUBLISH

Desktop Publishing – kurz DTP – gewinnt immer mehr an Bedeutung. Ob Sie originelle Visitenkarten oder professionelle Club-Zeitschriften drucken wollen: In diesem Workshop lernen Sie von den ersten Vorbereitungen bis zum druckreifen Ergebnis Schritt für Schritt den Umgang mit »Giga-Publish«, dem sensationellen DTP-Programm aus Sonderheft 39.

Fast unglaublich ist es, was das Programm »Giga-Publish« alles aufs Papier zu zaubern vermag. Die Druckqualität ist dabei so hoch, daß sich ohne weiteres auch anspruchsvollere Aufgaben bewältigen lassen.

Um »Giga-Publish« in allen seinen Fähigkeiten auszureizen, benötigt man aber einiges an Übung. Anhand zweier interessanter Beispiele lernen Sie in diesem Workshop, welche Arbeitsschritte beim Entwurf einer Druckseite anfallen.

Wenn Sie diesen Exkurs in die Welt Druckerei am Schreibtisch mitgemacht haben, verfügen Sie über genügend Wissen und Erfahrung, um eigenhändig so großartige Druckseiten wie die aus Bild 7 zu erstellen.

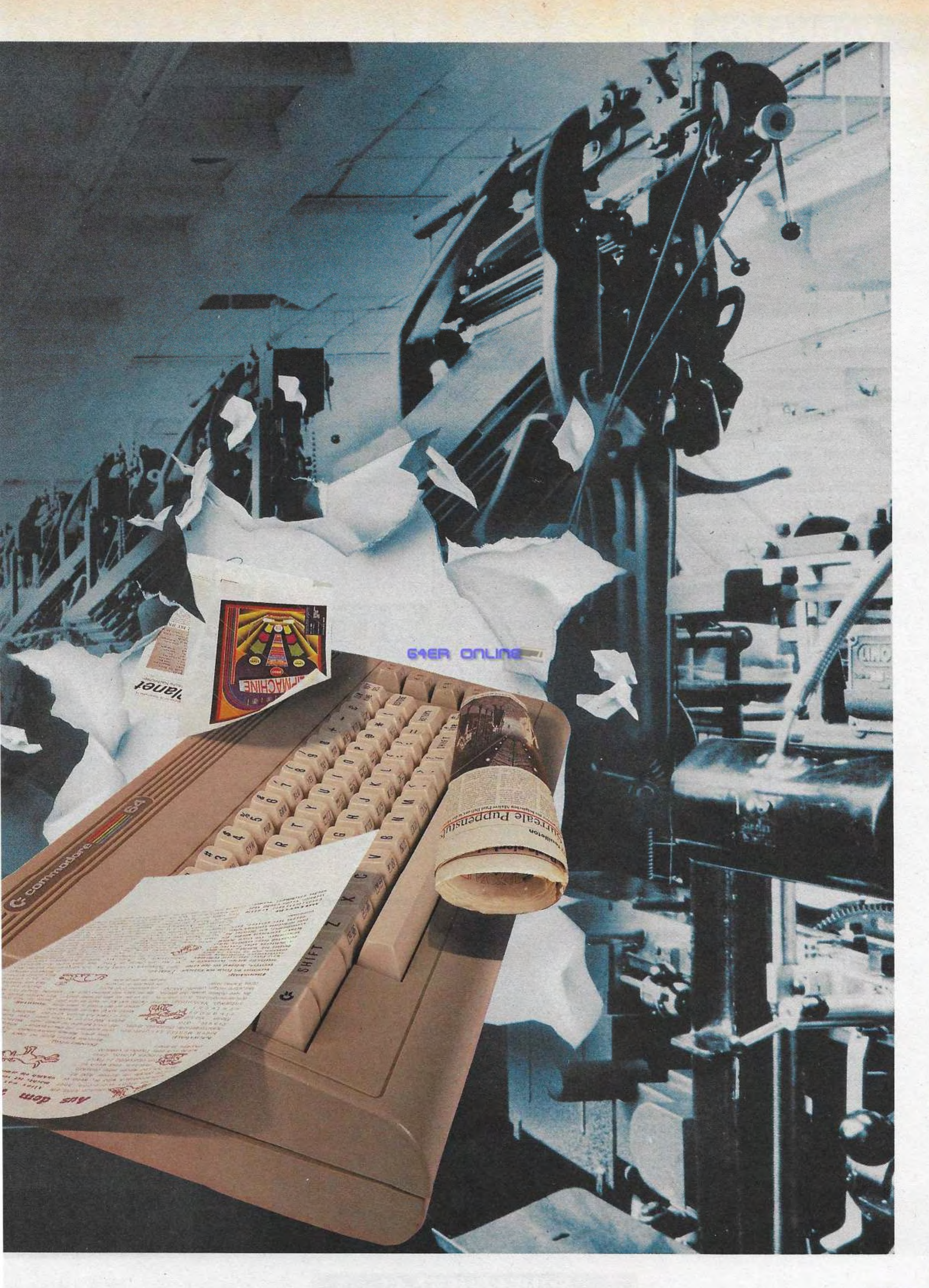
1. Ein Begleitnotiz-Zettel

Ein Begleitnotiz-Zettel wie in Bild 1 wird am Ende des Workshops nur noch eine Kleinigkeit für Sie sein. Mit diesem Beispiel wollen wir auch gleich anfangen. Da wir keine Bilder und nur einen Zeichensatz verwenden, ist der Aufwand recht gering.

Als Vorlage dienen die vielen Zettel dieser Art, wie sie in der Geschäftspost größerer Firmen üblich sind. Unser selbstgedrucktes Exemplar wirkt allerdings nicht minder professionell (vorausgesetzt, Sie ersetzen den Donald Duck durch Ihren Absender).

Bevor Sie alle folgenden Schritte des Workshops nachvollziehen, sollten Sie sich eine Arbeitsdiskette erstellen, auf der alle Programme von »Giga-Publish« enthalten sind. Alle weiteren Beispiele in diesem Workshop beziehen sich immer auf diese Arbeitsdiskette.




commodore 64
64ER ONLINE

Donald Duck
Erpelweg 17a
0000 Entenhausen 00
Tel: 000/000 00 00

Donald Duck, Erpelweg 17a, 0000 Entenhausen 00

Ich übersende beiligendes Schriftstück

- ☐ In Erledigung Ihres Schreibens
- ☐ gemäß telefonischer Absprache
- ☐ mit Dank zurück
- ☐ als Irläuter

mit der Bitte um

- ☐ Kenntnisnahme
- ☐ Behandlung wie besprochen
- ☐ Überprüfung
- ☐ Erledigung
- ☐ Rückgabe
- ☐ Verbleib

Mit freundlichen Grüßen

Datum Unterschrift

Bild 1. Der fertige Begleitnotiz-Zettel

1. Schritt: Der Zeichensatz

Als erstes sollten Sie eine ungefähre Vorstellung vom Endergebnis haben, die Sie in einer Skizze festhalten. Davon ausgehend kümmert man sich zunächst einmal um die Gestaltungsmittel, also Zeichensätze und Grafiken.

In unserem ersten Beispiel können wir das Problem »Grafiken« erst einmal vergessen, denn Bilder sind auf einer professionellen Begleitnotiz nicht nötig (höchstens im Rahmen eines Firmen-Logos).

Um dem Begleitzettel ein übersichtliches Äußeres zu geben, verwendet man sinnvollerweise nur einen einzigen Zeichensatz, dafür aber in verschiedenen Schriftgrößen und -dicken. Besonders geeignet für unser Beispiel ist eine kleine, unauffällige Schrift. Einen solchen Zeichensatz finden Sie übrigens unter dem Namen »f)0000« zu »Giga-Publish« gleich mitgeliefert.

Verfügen Sie nicht über einen passenden Zeichensatz im Giga-Publish-Format (und das dürfte wohl meistens der Fall sein), tritt erst einmal der Font-Konverter in Aktion.

Sie können auf das gesamte Angebot von Printfox-Zeichensätzen zurückgreifen, die inzwischen als Public Domain erhältlich sind. Dazu kopieren Sie sich den Zeichensatz im Fremdformat auf die Arbeitsdiskette. Anschließend lädt man dann den Font-Konverter mit

```
LOAD "FONT-KONVERTER",8,1 <RETURN>
```

Legen Sie nun die Diskette mit den Fremdzeichensätzen ins Laufwerk ein. Den Zeichensatz, den Sie konvertieren wollen, laden Sie mit Menüpunkt <b> in den Speicher. Unter Menüpunkt <a> (bearbeiten) läßt sich der Zeichensatz noch einmal ansehen. Die Größe der Zeichen können Sie an dieser Stelle festlegen (Bild 2).

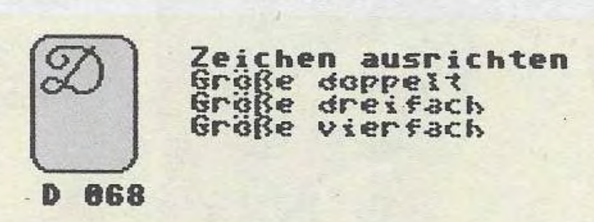


Bild 2. Der Font-Konverter

Den Zeichensatz, soweit überhaupt möglich, zu vergrößern, ist in der Regel nicht sinnvoll. Nur wenn Sie extrem große, plakative Schriften verwenden wollen, ist eine Vergrößerung zu empfehlen. Ansonsten speichern Sie den Zeichensatz nach dem Ausrichten mit Menüpunkt <c> auf die Arbeitsdiskette ab. Als Name dient eine vierstellige Nummer, die Sie sich merken sollten. Nach dieser Prozedur liegt der Zeichensatz im Giga-Publish-Format auf Diskette vor.

Sollten Sie bei einem späteren Ausdruck feststellen, daß der Text ohne Leerzeichen gedruckt wird, ist noch ein kleiner Zwischenschritt zu erledigen. Im Editor muß die Breite der Leerzeichen neu festgelegt werden. Anschließend werden die Texte korrekt gedruckt.

Der nächste Arbeitsschritt schließt sich direkt an: Der Zeichensatz besitzt noch nicht die richtigen Zeichen. Denn für unseren Begleitzettel benötigen wir zwei Sonderzeichen, die nicht im Standard enthalten sind. Dies wären einmal die rechteckigen Kästchen, in denen die entsprechende Mitteilung angekreuzt werden soll. Außerdem existiert selten ein breiter Unterstrich, mit dem sich eine waagrechte Linie zusammenstellen läßt (in unserem Beispiel die

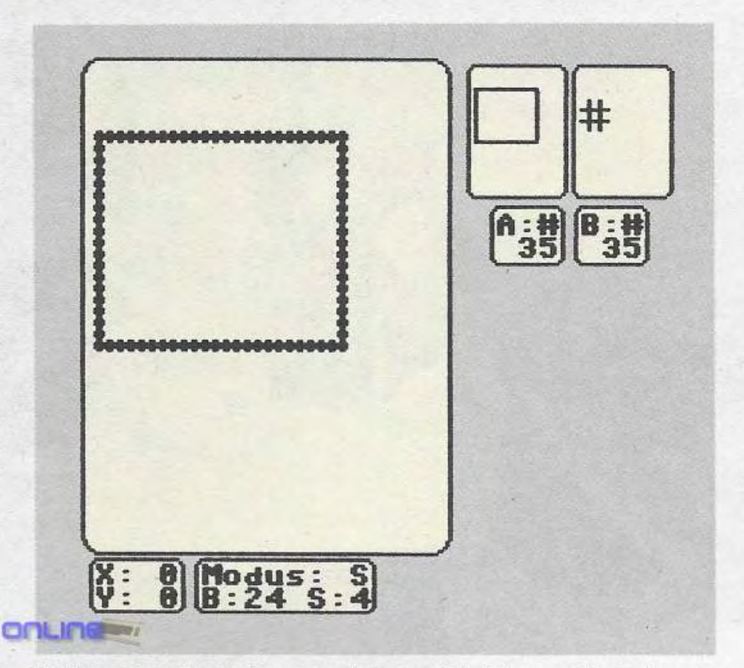


Bild 3a. Das Doppelkreuz wird zum Kästchen geändert

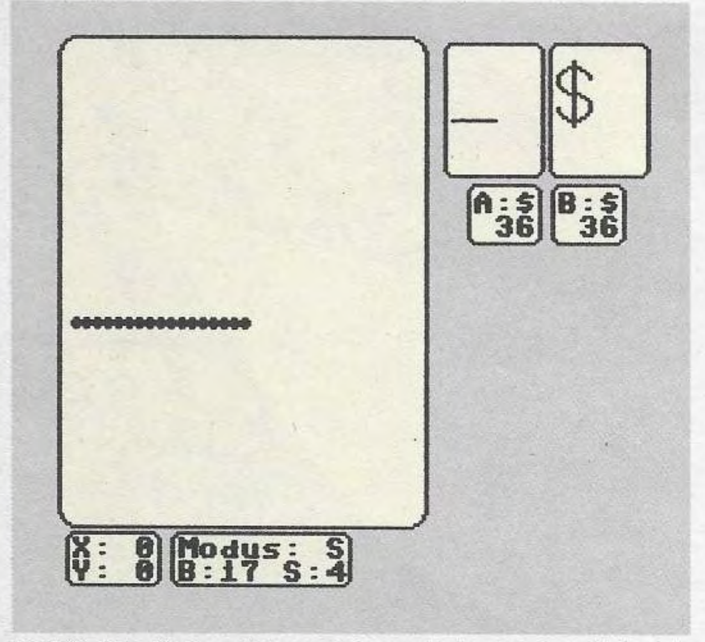


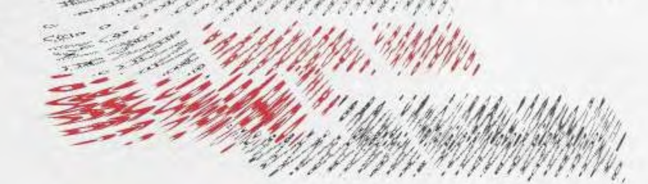
Bild 3b. Das Dollarzeichen wird zum Unterstrich geändert

Schriftlinie für Datum und Unterschrift). Der Zeichensatz muß bearbeitet werden.

Laden Sie gleich als nächstes den Zeichensatz-Editor mit

```
LOAD "GIGA-EDIT",8,1 <RETURN>
```

Wiederum mit Menüpunkt <b> laden Sie sich nun den zu editierenden Zeichensatz zweimal ein: einmal als Zei-



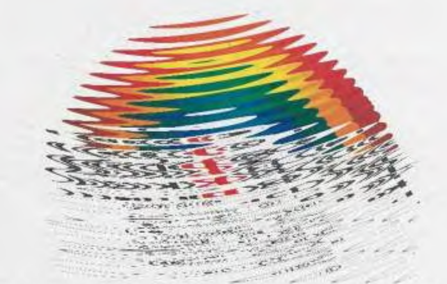




Bild 4. Jeder Textabschnitt erhält seine eigene Box

chensatz (a) und einmal als Zeichensatz (b). Zeichensatz (a) läßt sich dann beliebig verändern, während man mit Zeichensatz (b) immer die Originalversion zum Vergleich hat.

Zur Veränderung suchen Sie sich aus dem Zeichensatz zwei Zeichen, die in dem Projekt bestimmt keine Verwendung finden. Beispielsweise könnten Sie dazu das Doppelkreuz <#> und das Dollarzeichen <$> nehmen.

Das Doppelkreuz verändern Sie zu einem rechteckigen Kästchen und das Dollarzeichen zu einem breiten Unterstrich (Bilder 3a und 3b). Die zahlreichen Hilfsfunktionen, die das Editieren beträchtlich vereinfachen, sind ausführlich in Sonderheft 39 beschrieben. Wenn der Zeichensatz Ihren Vorstellungen entsprechend verändert wurde, speichern Sie ihn unter einem neuen Namen wieder auf die Arbeitsdiskette.

2. Schritt: Das Layout

Das Ergebnis bisher ist ein geeigneter Zeichensatz mit neuer vierstelliger Nummer auf der Arbeitsdiskette. Da unser Begleitzettel mit relativ wenig Text auskommt, verwenden wir zur Texteingabe keine externe Textverarbeitung, sondern den in »Giga-Publish« integrierten komfortablen Editor.

Zunächst aber kommt ein sehr wichtiges Kapitel an die Reihe, nämlich das Layout. Wo früher noch mit Papier und Schere herumgebastelt wurde, genügt uns bei »Giga-Publish« der Joystick oder die Maus.

Die erste Überlegung beim Layouten ist immer die Frage: Wie viele Textspalten werden benötigt?

Da die Begleitnotiz deutlich in eine linke Hälfte (Absender und Freiraum für Adresse) und eine rechte Hälfte (Informationen und Unterschrift) aufgeteilt ist, fällt die Entscheidung leicht: zwei Spalten.

Klicken Sie also das dritte Icon von oben am Layout-Bildschirm einmal und nach einer Pause zweimal schnell an. Nun ist die Bildschirmseite in zwei Textspalten gegliedert.

Der Begleitzettel soll aber nur ein Drittel einer DIN-A4-Seite groß sein. Daher schiebt man (nach Aktivierung des oberen Move-Icons) den unteren Seitenbegrenzer nach oben, bis die Druckseite die gewünschte Größe einnimmt. Die rechte Textspalte wird, wie wir gleich sehen werden, eigentlich gar nicht gebraucht, weshalb wir sie zur Vereinfachung gleich löschen (Abfalleimer-Icon). Gleichzeitig schieben wir den rechten Rand der übriggebliebenen Spalte ein bis zwei Punkte nach links, um den Freiraum in der Mitte der Seite zu vergrößern (dies hat lediglich kosmetische Funktion).

Das Problem der unterschiedlich angeordneten Zeilen auf der rechten Hälfte der Seite lösen wir am einfachsten durch Boxen: Jeder »Abschnitt« erhält seine eigene Box (Bild 4). Auf diese Weise gibt es keine Schwierigkeiten mit den parallelen Spalten, in denen die anzukreuzenden Informationen angeordnet sind. Außerdem lassen sich die Abschnitte dann später noch leicht nach oben oder unten verschieben (wenn zum Beispiel manche Zeilen zu nahe aneinanderkleben).

Wir benötigen für den Begleitnotiz-Zettel insgesamt sieben Boxen, wobei die oberen Boxen am einfachsten Kante an Kante angeordnet werden (Bild 5). Damit ist das grafische Layout eigentlich schon beendet. Allerdings sind noch einige andere Punkte im Layout-Menü zu bearbeiten.

Da wäre zunächst Menüpunkt »d) Fonts«. Da wir auf dem Begleitzettel nur einen Zeichensatz verwenden wollen, geben Sie nur für den Zeichensatz Nummer Null den Namen des vorher konvertierten und veränderten Zeichensatzes

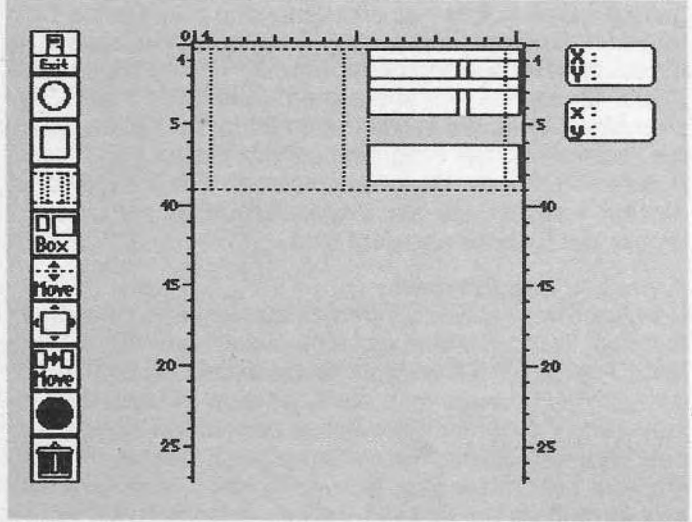


Bild 5. Das Layout des Begleitzettels

ein. Versichern Sie sich, daß vor dem Druck auf <RETURN> dieser Zeichensatz auch auf der Arbeitsdiskette im Laufwerk zu finden ist.

Zum Abschluß wählen wir die Extras aus Menüpunkt »e)« an. Hier erfahren Sie, daß auf der Seite 0 eine Textspalte und sieben Boxen existieren. Den nächsten Wert, nämlich den Zeilenabstand in den Textspalten, sollten Sie verändern, da sonst die Zeilen zu nah aneinanderkleben. Nach mehrmaligem Ausprobieren stellt sich hier der Wert 18 als besonders geeignet heraus.

Die restlichen Informationen sind für unser relativ einfaches erstes Beispiel nicht zu verändern. Die Boxen haben allesamt den Zeilenabstand 0, keinen Rahmen und kein Bild. Für den Ausdruck von Bild 4 habe ich übrigens lediglich für jede Box den Rahmen eingeschaltet.

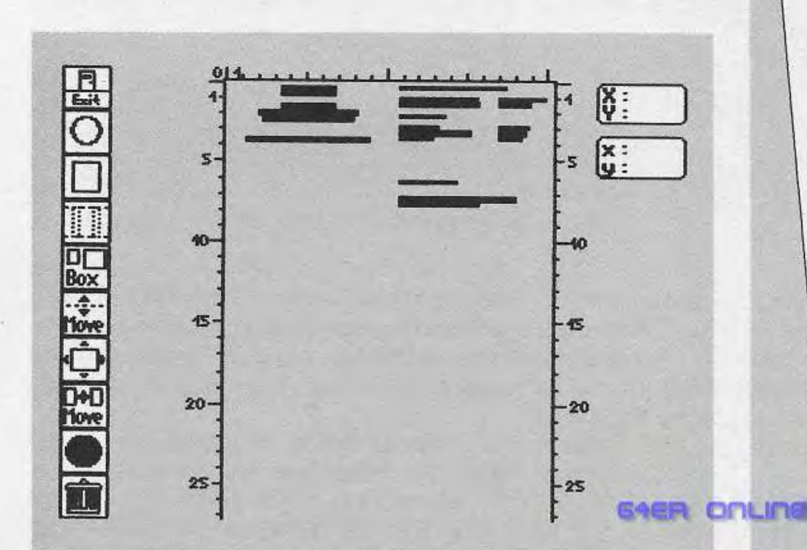


Bild 6. Das Preview des Begleitzettels

3. Schritt: Der Text

Bei der Eingabe des Textes mit dem Texteditor verwenden wir Steuerzeichen, wie es im Sonderheft 39 beschrieben ist. So werden die Absender auf der linken Hälfte zentriert und in variabler Größe und Schriftstärke verwendet. Die Boxentexte werden dann linksbündig und teilweise halbfett gedruckt. Außerdem sollten Sie noch beachten, daß die umdefinierten Sonderzeichen (Kästchen und Unterstrich) im Editor durchaus noch als Doppelkreuz und Dollarzeichen gehandhabt werden. Die neue Definition kommt erst beim Ausdrucken zum Vorschein. Um einen Begleitzettel wie in Bild 1 zu erhalten, geben Sie Listing 1 ein und speichern den Text auf der Arbeitsdiskette. Laden Sie ihn anschließend in den Editor, um sich in Ruhe die verwendeten Steuerzeichen anzusehen. Alle geänderten Zeichensätze sowie die verwendeten Bilder und Texte finden Sie komplett auf der Programmservice-Diskette zu diesem Sonderheft. Für die Bilder beispielsweise fehlt der Platz in diesem Heft, da sie als Grafikinformation mit zirka 32 Blocks auf Diskette abgelegt sind.

4. Schritt: Das Preview

Bevor Sie sich nun an den zeitaufwendigen Ausdruck machen, sehen Sie sich erst einmal das vorläufige Ergebnis im Preview an. Obwohl die Textzeilen nur als Balken dargestellt sind, erkennt man doch, wo zum Beispiel die Formatierung nicht stimmt, wo Zeilen zu nah aufeinander kleben, oder wo Texte nicht in die Box passen. Sollten zum Beispiel die Texte in den Boxen C und F nicht in eine Zeile passen, so muß die Box im Layout etwas vergrößert werden. Im Bild 6 sehen Sie das korrekte Preview.



Einla
zum Jahrestreffen
K.., des Verbandes
der Computer-Kot
uns über Ihren

Sicherlich kennen Sie die Situation: Gerade haben Sie das beste Programm Ihres Lebens geschrieben, und dann stürzt der Computer ab. Schuld daran war mit Sicherheit ein Computer-Kobold, der die Bits und Bytes kräftig durcheinander gebracht hat!

Kobolde!!!

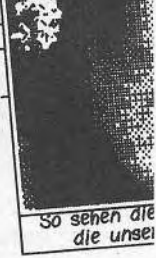
Da schon eine Unzahl von Computer-Freaks auf diese Weise in den Wahnsinn getrieben wurden, wurde der V. z. B. d. C. K. gegründet. Unser erklärtes Ziel ist die Ausfindigmachung und Ausrottung der Kobolde.
Akute Aufgabe unseres Vereines ist es, die Forschungen auf dem Gebiet der Computer-Koboldologie vehement vorwärts zu treiben. Denn immernoch wissen wir viel zu wenig über diese Geißel der Menschheit.
Doch einige bedeutende Fakten konnten schon herausgefunden werden. Über diese Erkenntnisse wollen wir Sie kurz informieren:

Wie verbreiten sich die Kobolde?

Aller Wahrscheinlichkeit nach geschieht dies recht hinterhältig - nämlich über die nationalen Stromnetze.

in Laptops!

Wer also seinen Computer an eine Steckdose anschließt, ist schon akut gefährdet.
Das beste Mittel gegen die zahlreichen, von Kobolden verursachten Abstürze wäre also, den Computer nicht einzuschalten. Jedoch zeigten sich nach Aussagen unserer Forschungsabteilung bei dieser Methode einige unangenehme Nebenwirkungen
Schon seit einiger Zeit reagiert die Industrie auf die gemeinen Netzkabel-Kobolde durch die verstärkte Herstellung von Laptop-Modellen. Allerdings schei-
nen sich einige Kobolds-Mu
tanten auch auf Portable
Batterien spezialisiert z
haben.
Bisher noch nicht einwanc
frei beweisen konnte ma
die Thesen, Laptop-Kc
bolde würden auch de
Kontrast der LCD-Schirme niedrig halten sowie Spiegelungen und Reflexionen begünstigen.



So sehen die
die unser

Vermehren sich d
Kobolde?

Selbstverständlich ver
ren sich die Compute
bolde ganz beträch
Zwar ist ihr Sexual
noch nicht einwandfr
klärt, eines steht all
fest: Kobolde verm
sich schneller als C
ter. Denn wer kann t
ten, daß es heute
Amiga-Ära wesentlicl
Computer-Abstürze
also zu Zeiten de
oder gar VC 20.
Interessanterweise
wir den Namen ei
bold-Typs, er ner
selber »Guru«. Zud
die Individuen dies
durchnummeriert.
schäftigung der »(
bolde ist übrigens
tation.

dung

des V. z. B. d. C. zur Bekämpfung
olde. Wir würden
Besuch freuen.

Welche Schutzmaßnahmen gibt es bisher?

Leider nicht sehr viele. Einen interessanten Weg hat aber die Weltfirma IBM beschritten. Sie versucht es mit dem sogenannten »Micro-Channel«. Diese Schnittstelle soll, wie der Name schon sagt, so klein sein, daß kein Kobold hindurch paßt. Eindeutige Erfolge konnten noch nicht erreicht werden.

Ein anderer Weg wird zur Zeit von mehreren Firmen erprobt. Hierbei findet das geflügelte Wort »alter Kobold ist doch kein D-Zug« Verwendung. Die Computer sollen so schnell getaktet werden, daß der Kobold sich zwischen den herumflitzenden Bytes nicht mehr auskennt und daher auch keinen Schaden anrichten kann. Jedoch auch bei 43 Mhz zeigt sich noch keinerlei gesteigerte Absturzsicherheit.

gefährlichen Kobolde aus, e Computer bewohnen.

Gibt es also gar keine Hilfe?

Doch, einige Lichtblicke sind durchaus zu sehen. Eine etwas brutale Methode soll sich zum Beispiel bewährt haben. Durch schnelle Drucker-Interfaces werden die Kobolde nämlich über den seriellen oder parallelen Port aus dem Computerinneren geschleudert. Kurz darauf landen sie im Druckerkopf und fliegen auf das Papier. Je mehr Nadeln der Drucker nun hat, desto höher ist die Wahrscheinlichkeit, daß der Kobold von einer Nadel getroffen wird.

Wundern Sie sich also nicht über verpatzte Hardcopies – es liegt nicht am Drucker! Merke: Regelmäßig den Druckkopf von Blutspuren reinigen.

Wesentlich Drucker-schonender wird den Kobolden auf eine andere Art der Garaus gemacht. Die ganze Welt arbeitet zur Zeit an der Entwicklung sogenannter »Computer-Viren«. Diese häßlichen kleinen Tierchen sollen in Zukunft auf Kobolde abgerichtet werden und diese für immer ausrotten. Bedingung für den Erfolg: die Viren müssen sich in sämtliche Datensysteme der Welt ausbreiten!



Ein von Kobolden befallener Rechner.

VIREN HELFEN

Wenn Sie an diesem hochbrisanten Thema interessiert sind, besuchen Sie uns doch auf der Jahresversammlung des V. z. B. d. C. K.III Neben hochinteressanten Informationen und der Möglichkeit zum Gespräch mit Fachleuten erhalten Sie als Geschenk einen Pumuckl-Schlüsselanhänger.

Bild 7. Die fertige Einladung

5. Schritt: Der Ausdruck

Sind im Preview keine Fehler zu entdecken, ist alles für einen ersten Ausdruck bereit. In diesem ersten kleinen Beispiel ohne Bilder und mit nur einem Zeichensatz geht dies verhältnismäßig schnell und nimmt nur zirka vier Minuten in Anspruch. Anhand dieses Ausdrucks sehen Sie schnell, ob kosmetische Änderungen nötig sind.

In unserem Beispiel empfiehlt es sich auch, die Größe des Ausdrucks mit den Maßen eines entsprechenden Kuverts zu vergleichen. Dabei sollten Sie darauf achten, daß der kleine unterstrichene Absender im Sichtfenster des Kuverts erscheint.

Wenn nun alles zu Ihrer vollen Zufriedenheit gelöst ist, so können Sie Layout und Text (nicht vergessen!) abspeichern, um sich bei Bedarf (oder wenn Sie einmal viel Zeit übrig haben) neue Begleitnotizen auszudrucken.

2. Eine Einladung

Das zweite Beispiel, das wir mit »Giga-Publish« erarbeiten wollen, ist eine aufwendig gestaltete Einladung (Bild 7). Diese Einladung soll neben dem mehrspaltigen Fließtext auch Überschriften, Vor- und Nachspänne sowie Bilder mit Bildunterschriften enthalten. Alles in allem also eine »bunte« und vielseitige Demonstration.

1. Schritt: Die Zeichensätze

Natürlich verwendet man auf einer Einladung, die ansprechend wirken soll, nicht nur einen Zeichensatz. Allerdings muß man als »Gestalter« einen goldenen Mittelweg finden. Denn verschiedene Zeichensätze machen das Gesamtbild zwar abwechslungsreich, doch bei zu vielen Zeichensätzen wirkt die Vielfalt verwirrend und unprofessionell.

Als Zeichensatz für den Fließtext benutzen wir wieder den »f)0000«, der im Sonderheft 39 mit Giga-Publish abgedruckt wurde. Dies ist ein einfacher und nicht zu großer Zeichensatz. Ein ähnlicher wird ja auch in dieser Zeitschrift verwendet. Vor- und Nachspann erscheinen üblicherweise im gleichen Zeichensatz wie der Fließtext, wenn auch in einer anderen Dicke.

Abwechslung im Text erzeugen wir mit Überschriften. Sinnvollerweise verwenden wir für die vier Zeilen unter »Einladung« einen typischen computerlesbaren Zeichensatz. Dies ist sofort ein optisches Signal, um welches Thema es sich handelt.

Ein interessantes Mittel zur Auflockerung unseres Textes sind die Zwischenüberschriften (»Kobolde!!!«, »in Laptops«, »Viren helfen«). Sie fassen kurz und prägnant das Thema eines Textabschnittes zusammen. Wir plazieren sie irgendwo innerhalb dieses Textabschnittes. Auch in dieser Zeitschrift werden solche Zwischenüberschriften verwendet, Sie erkennen sie an dem Balken darunter (eine dünne und eine dicke Linie).

Die textbezogenen Überschriften (»Wie verbreiten sich die Kobolde?« und so weiter) bekommen der Übersichtlichkeit halber einen einheitlichen Zeichensatz, der sich in der Dicke deutlich vom Fließtext abhebt. So groß wie die Zwischenüberschriften sollte er allerdings nicht sein, da wir sonst keinerlei Abwechslung erzielen. Schließlich benötigen wir noch einen sehr kleinen, aber gut lesbaren Zeichensatz für die Bildunterschriften.

Konvertieren Sie sich also die Zeichensätze Ihrer Wahl ins Giga-Publish-Format und speichern Sie sie auf die Arbeitsdiskette.

2. Schritt: Die Bilder

Auch die Grafiken, die nachher auf der Einladung erscheinen sollen, müssen zuerst ins Giga-Publish-Format konvertiert werden. Dazu dient das Konvertierungs-Programm, welches Sie mit

```
LOAD "BILD-KONVERTER",8,1 <RETURN>
```

laden. Danach laden Sie mit Menüpunkt »b)« die Grafik im Fremdformat ein. Verwenden Sie bei normalen Grafiken immer Bereich 1.

Theoretisch wäre es auch möglich, vier komplette Grafik-Bildschirme einzuladen und dann als ein einziges Bild zu verwenden. So ist zum Beispiel das Programm »Giga-CAD« in der Lage, seine Bilder mit 640 x 400 Punkten zu berechnen und in vier Bildschirmen abzuspeichern. Allerdings benötigen wir für unsere Einladung keine übergroßen Grafiken dieser Art.

Als nächstes kommt die Wahl des Ausschnitts. Am besten läßt man dabei keine leeren Ränder, sondern macht den Ausschnitt möglichst klein. Dann hat man nämlich oft noch die Möglichkeit, das Bild später vergrößert auszudrucken.

Ist der Ausschnitt festgelegt, sollten Sie unbedingt Menüpunkt »d) Info« aufrufen. Hier erfährt man zunächst einmal die Größe des Ausschnittes in Einzelpunkten. Sehr wichtig sind die Werte in Klammern bei »Druckbreite« und »Druckhöhe«. Diese Zahlen geben an, wie hoch und breit eine Box im Layout sein muß, um das Bild aufnehmen zu können. Außerdem bekommt man hier schon den ersten Eindruck von der Größe des Bildes im Ausdruck. Zum Vergleich: Der Layout-Bildschirm von Giga-Publish ist 161 Punkte breit und 194 Punkte hoch.

Durch die hohe Auflösung des Druckers, die ja von »Giga-Publish« voll unterstützt wird, geraten manche Bilder kleiner als erwartet. So nimmt eine normale Grafik von 320 x 200 Punkten weniger als ein Fünftel der Breite und weniger als ein Zehntel einer Druckseite in Anspruch. Daher vergrößern wir zu kleine Bilder (in unserem Beispiel ist das mit dem mittleren Bild, dem Frauengesicht, geschehen).

Zu diesem Zweck werden <A> und <B> je einmal gedrückt. Dadurch verdoppeln sich horizontale und vertikale Ausmaße. Giga-Publisch wird nun statt einem Punkt jeweils vier quadratisch angeordnete Punkte drucken, wodurch das Bild insgesamt viermal mehr Platz einnimmt. Sie können sowohl »Größe X« als auch »Größe Y« bis zu versechsfachen. Dabei sollten aber beide Größen jeweils den gleichen Wert erhalten, da sonst starke Verzerrungen entstehen.

Wenn die richtige Größe gewählt ist, notieren Sie sich am besten die Werte in den Klammern und speichern anschließend das Bild ab.

»X-Format« und »Y-Format« bestimmen, ob das Bild in der Box später zentriert oder bündig gedruckt werden soll. Meistens werden Sie hier wohl die Werte 0/0 für zentriert verwenden.

Noch etwas zur Bilderwahl: Es ist ein interessanter psychologischer Trick, auf der Mitte einer Seite ein menschliches Gesicht abzudrucken. Ein Gesicht wirkt nämlich unweigerlich als Blickfang und lenkt automatisch die Aufmerksamkeit des Lesers auf diese Seite.

3. Schritt: Der Fließtext

Der in Giga-Publish eingebaute Editor bietet zwar erstaunlich viel Komfort, doch für die Eingabe längerer Texte ist eine echte Textverarbeitung wesentlich angenehmer. Für diese Zwecke bietet sich das Programm »Master-Text 64« an, welches ebenfalls im Sonderheft 39 veröffentlicht wurde.

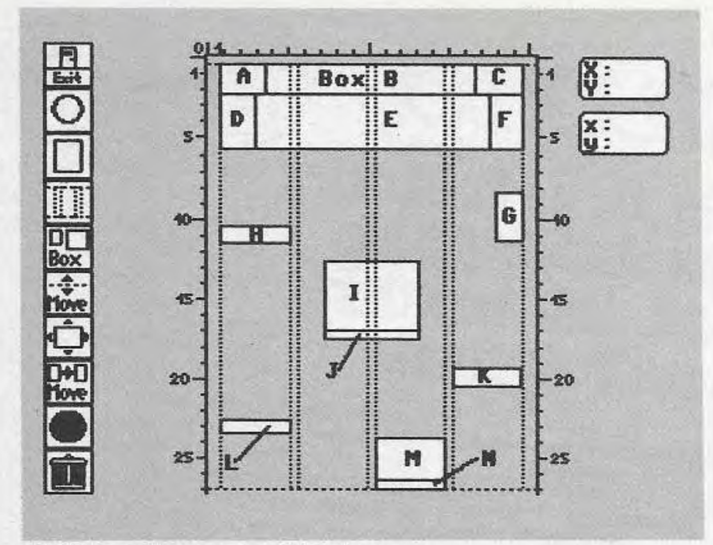


Bild 8. Das Layout der Einladung

Geben Sie aber lediglich den Fließtext ein, also nicht die großen (Zwischen-)Überschriften und die Bildunterschriften. Auch müssen Sie den Text ganz ohne Steuerzeichen eingeben, diese werden erst später mit dem Giga-Publish-Editor eingefügt. Den Text speichern Sie dann auf Ihrer Arbeitsdiskette zur weiteren Verarbeitung ab.

Nun tritt der Textkonverter in Aktion. Dieses Programm konnte aus technischen Gründen leider nicht im Sonderheft 39 abgedruckt werden, auf der dazugehörigen Leserservice-Disk ist es aber enthalten. Das Listing finden Sie im Sonderheft 40. Das Programm wird geladen mit

```
LOAD "MASTER/KONVERT",8,1 <RETURN>
```

Die Bedienung ist denkbar einfach. Man lädt den Text im Master-Text-Format ein und speichert in anschließend unter beliebigem Namen wieder auf Diskette. Nun liegt der Text im Giga-Publish-Format vor.

4. Schritt: Das Layout

Nun kommt man zum kompliziertesten Teil der Arbeit, dem Entwurf des Layouts. Hier ist es wieder nötig, eine genaue Vorstellung vom späteren Aussehen der Seite zu haben. Am Anfang steht immer die Entscheidung über die Spaltenzahl. In unserem Fall eignen sich vier Spalten am besten. Dadurch ist der recht umfangreiche Text deutlich aufgegliedert.

Die Spalten sind relativ schmal, so daß auch bei enger Schrift und kleinem Zeilenabstand keine Leseschwierigkeiten entstehen. Außerdem wird bei vier Spalten die Verwendung von mehreren Überschriften erleichtert. Optisch unangenehm ist es, wenn eine breite Spalte von einer nur schmalen Überschrift unterbrochen wird.

In Bild 8 finden Sie das passende Layout. Box B dient zunächst einmal zur Aufnahme des riesig gedruckten Wortes »Einladung«. Da der Schriftzug dennoch nicht die gesamte Seitenbreite in Anspruch nimmt, bleibt links und rechts davon noch genug Platz für die Einblendung der Schilder. Dazu werden selbstverständlich eigene Boxen (A und C) benötigt. Die kleinen Logos neben der Überschrift sind leider höher als die Schrift, daher müssen auch die Boxen A und C höher als Box C sein. Daß sie sich mit den darunterliegenden Boxen überlappen, ist nicht weiter schlimm.

Darunter folgt der erste Vorspann. Die Box E ist etwas breiter als Box B, dadurch wirkt die Überschrift über dem Vorspann wie eine »Krone«. Um den Vorspann aber dennoch kompakt und übersichtlich zu halten, nimmt Box E nicht die gesamte Seitenbreite in Anspruch. Die Boxen D und F sind unbedingt notwendig, damit der Platz seitlich

von Box E freigehalten wird. Wären sie nicht vorhanden, würde der Fließtext schon hier oben beginnen. Aus dem gleichen Grund liegen die Boxenränder auch genau auf den Spaltenrändern. Wäre hier auch nur eine Punktzeile frei, so würde »Giga-Publish« versuchen, dort noch Text unterzubringen.

Box G muß wieder groß genug sein, um das kleine Bild aufzunehmen. Zusätzlich wurde die Box noch etwas höher gemacht, um den Text etwas aufzulockern. Die Boxen H, K und L werden die Zwischenüberschriften enthalten. Die vertikalen Ränder müssen aus den schon oben genannten Gründen genau mit den Spaltenrändern zusammentreffen. Die genaue Höhe und Lage optimiert man dann nach den ersten Previews oder Ausdrücken.

Die Boxen I und J sind recht interessant, da sie genau zwischen zwei Spalten liegen. Vor allem muß natürlich das Bild in die Box I passen. Achten Sie aber in solchen Situationen darauf, daß links und rechts von der Box genug Platz für den Text bleibt. Mindestens ein Wort muß nämlich immer noch in eine Zeile passen, sonst gibt es Probleme.

Die Boxen M und N liegen dann wieder mit ihren Rändern auf den Spaltenrändern, somit ergeben sich keine Schwierigkeiten. Übrigens wurden der obere und untere Seitenbegrenzer ein beziehungsweise zwei Punkte nach oben beziehungsweise unten verschoben, um auf der Seite noch etwas mehr Platz zu bekommen.

Unter Menüpunkt »d) Fonts« des Layout-Menüs gibt man nun die Namen der konvertierten Zeichensätze ein. Achten Sie darauf, daß dabei eine Diskette mit den entsprechenden Zeichensätzen im Laufwerk liegt. Denn »Giga-Publish« lädt sofort die Font-Dateien nach, die mit »d)« beginnen (beispielsweise »d)0000«). In diesen kurzen Dateien sind die Höhe und die Breiten der einzelnen Zeichen des Zeichensatzes gespeichert. Diese benötigt »Giga-Publish« zur Berechnung des Previews und des Ausdrucks.

Wichtig ist auch die Eingabe der Extras unter Menüpunkt »e)«. Den Zeilenabstand in den Textspalten lassen Sie am besten auf dem Wert 1. Der Auto-Center 0 besagt, daß ein einzelnes Wort in einer Zeile linksbündig gedruckt wird. Bei Microspace 5 können zwischen den Buchstaben eines Wortes höchstens fünf leere Punktspalten eingefügt werden (nur bei Blocksatz nötig). Die komplette Auflistung der Extras für die Einladung finden Sie in Tabelle 1.

Die Extras zum Layout der Einladung

a)	Layout	0
	Spalten	4
	Boxen	14
b)	Zeilenabstand	1
c)	Auto-Center	0
d)	Microspace 5	

Boxen:

Nr.	ZAB	Rahmen	Bild	Bild-Nr.	Bild-Name
A	0	n	j	0	logo
B	0	n	n		
C	0	n	j	0	logo
D	0	n	n		
E	0	n	n		
F	0	n	n		
G	0	n	j	0	logo
H	0	n	n		
I	0	j	j	1	face
J	0	j	n		
K	0	n	n		
L	0	n	n		
M	0	j	j	2	comp
N	0	j	n		

Tabelle 1. Die Werte des Extra-Menüs

5. Schritt: Die Textbearbeitung

Nachdem Sie das Layout abgespeichert haben, aktivieren Sie den Editor. Dort laden Sie den konvertierten Fließtext ein. Nun werden sämtliche Steuerzeichen für Textdicke, -format und -größe sowie die Boxen-Texte eingegeben. Einige Dinge sind dabei zu beachten, die man leicht vergißt.

Jeder Sondermodus (fette Schrift, Unterstreichung) muß am Ende des betreffenden Textabschnittes wieder ausgeschaltet werden! Vergessen Sie nicht die Angabe des Zeichensatzes am Fließtext-Anfang! Boxentexte müssen durch die entsprechenden Sonderzeichen umschlossen sein (Anfang: <–> <X> <Name der Box>, Ende: <–> <X> <X>)!

Der eigentliche Fließtext wird natürlich im Blocksatz formatiert, das wirkt sehr professionell. Zwei Ausnahmen existieren aber: Der Text links von Box I wird linksbündig formatiert, der Text rechts von Box I rechtsbündig. Dies hat einen einfachen Grund. Passen in die verschmälerte Spalte nur noch ein oder zwei Worte, so werden diese so weit gestreckt, bis der vorhandene Platz ausgefüllt ist. Dies sieht nicht besonders vorteilhaft aus. Daher formatiert man so, daß der Spaltenrand »glatt« bleibt und der Text zur Box hin einen unregelmäßigen Rand bildet. An welcher Stelle im Fließtext die Steuerzeichen eingefügt werden müssen, weiß man allerdings erst nach dem ersten Ausdruck.

Vor- und Nachspann sowie Über- und Unterschriften zentriert man, um sie optisch zu betonen und Abwechslung in den Text zu bringen.

Nun haben Sie die Chance, die optische Qualität des Ausdruckes ganz bedeutend zu steigern. Denn in so schmalen Spalten wie in unserem Layout passiert es oft, daß das jeweils letzte Wort gerade nicht mehr in die Zeile paßt und deshalb in die nächste Zeile »umgebrochen« wird. Dies hat aber bei Blocksatz den Nachteil, daß Wörter und Leerzeichen unschön gestreckt und die Zeichenabstände sehr unregelmäßig werden. Bei linksbündiger Formatierung kommt es zu sehr zackigen Rändern.

Daher bietet »Giga-Publish« die Möglichkeit, Trennvorschläge einzufügen. Paßt ein Wort nicht mehr komplett in eine Zeile, so wird automatisch bei einem Trennvorschlag getrennt. Je mehr Trennvorschläge der Text also enthält, desto optimaler kann getrennt werden. Und desto schöner ist dann auch das Schriftbild.

Nehmen Sie sich also ruhig die Zeit, möglichst viele Trennvorschläge (mit <CTRL->) einzufügen. Letztendlich paßt dadurch übrigens auch mehr Text auf eine Seite.

6. Schritt: Das Preview

Wenn Sie auch den Text abgespeichert haben, ist es Zeit für das erste Preview. Lassen Sie sich immer Spalten, Boxen und Bilder zusammen ausgeben, auch wenn dies relativ lange dauern kann. Nicht vergessen: Zur Ausgabe des Previews muß erst noch das Seitennummern-Symbol angeklickt werden!

Hier erkennt man schon ganz gut, was wo stehen wird. Achten Sie darauf, daß Zwischenüberschriften nicht in den Vorspann fallen oder zu nahe an einer Fließtext-Überschrift kleben. Solche Mängel können durch Verschiebung der Boxen im Layout behoben werden.

Fließtext-Überschriften sollten weder seitlich von einer Box noch am unteren Spaltenende stehen. Das wirkt nicht gut. Fügen Sie nach Bedarf Leerzeilen (Carriage Returns) in den Text ein.

Am Preview erkennt man auch, ob die Formatierung überall stimmt. Text im Blocksatz wird immer mit glattem linken und rechten Rand angezeigt (außer den Absätzen, die sind zu erkennen). Um die Aufteilung zwischen Spaltentext und Boxen genauer erkennen zu können, schalten Sie einfach im Extras-Menü alle Boxen-Rahmen ein.

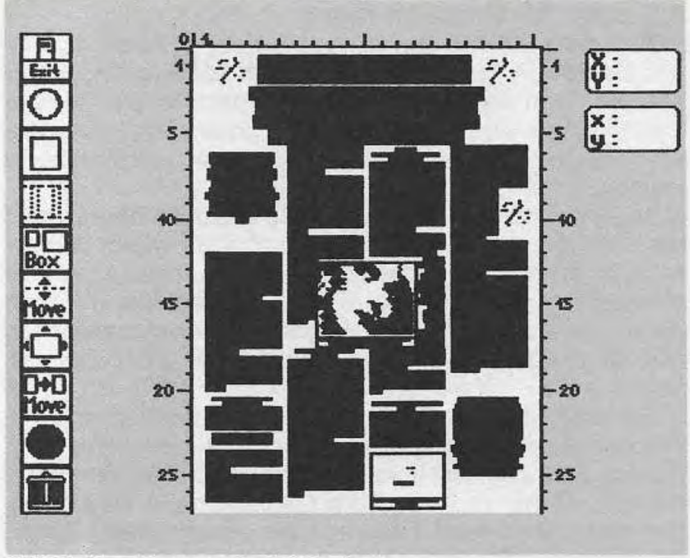


Bild 9. Das Preview der Einladung

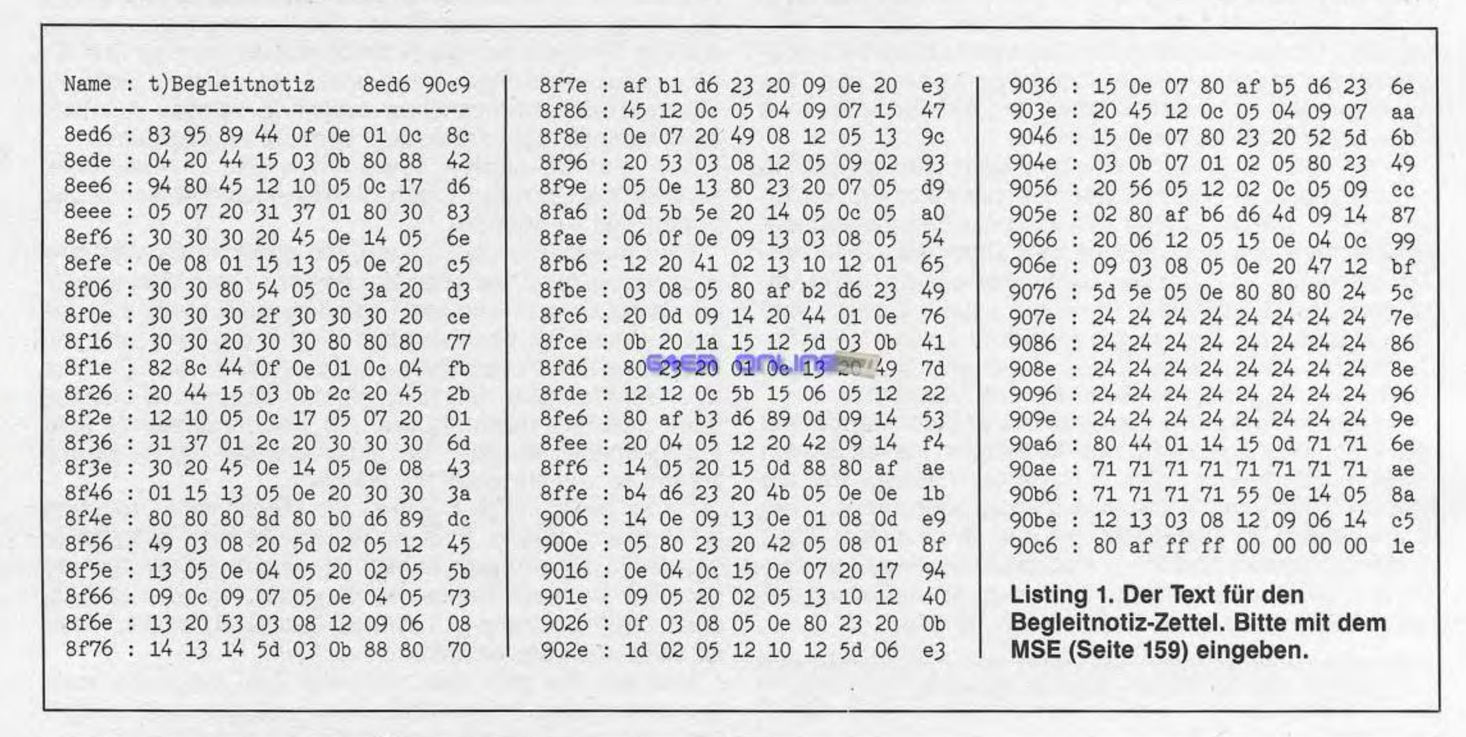
Nach den Korrekturen sieht das Preview wie in Bild 9 aus. Vergessen Sie nicht, Text und Layout zu speichern!

7. Schritt: Der Ausdruck

Während des Ausdrucks sollten Sie sich eine Kaffee-Pause gönnen. Bis sich die komplette Seite beispielsweise aus einem Star NL-10 gequält hat, vergeht nämlich über eine halbe Stunde. Trotzdem sollten Sie in der Nähe des Druckers bleiben. Manche gravierende Fehler fallen nämlich schon während des Druckens auf, und dann freuen sich Farbband und Mitbewohner, wenn Sie den Ausdruck vorzeitig abbrechen (mit <RUN/STOP>).

Im Ausdruck sieht man deutlich, wo zum Beispiel seitlich der Box umformatiert werden muß. Auch fallen einige Stellen ins Auge, an denen ein Trennvorschlag guttun würde. Auch lohnt es sich meistens, die Rechtschreibung noch einmal zu kontrollieren.

Nach all diesen letzten kleinen Änderungen aber haben Sie den perfekten Ausdruck einer Einladung zur Jahresversammlung des V. z. B. d. C. K.! Übrigens — wollen Sie nicht auch beitreten? (Nikolaus Huber/ef)

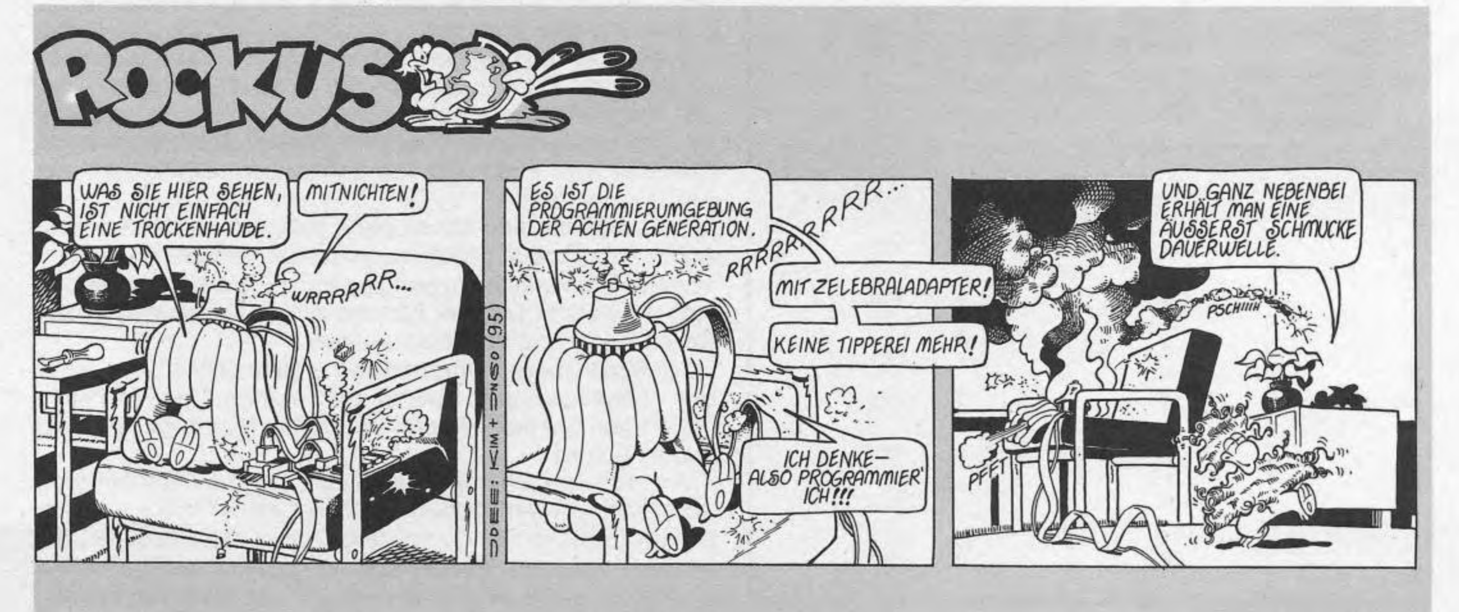


```
Name : t)Begleitnotiz     8ed6 90c9
-----------------------------------
8ed6 : 83 95 89 44 0f 0e 01 0c  8c
8ede : 04 20 44 15 03 0b 80 88  42
8ee6 : 94 80 45 12 10 05 0c 17  d6
8eee : 05 07 20 31 37 01 80 30  83
8ef6 : 30 30 30 20 45 0e 14 05  6e
8efe : 0e 08 01 15 13 05 0e 20  c5
8f06 : 30 30 80 54 05 0c 3a 20  d3
8f0e : 30 30 30 2f 30 30 30 20  ce
8f16 : 30 30 20 30 30 80 80 80  77
8f1e : 82 8c 44 0f 0e 01 0c 04  fb
8f26 : 20 44 15 03 0b 2c 20 45  2b
8f2e : 12 10 05 0c 17 05 07 20  01
8f36 : 31 37 01 2c 20 30 30 30  6d
8f3e : 30 20 45 0e 14 05 0e 08  43
8f46 : 01 15 13 05 0e 20 30 30  3a
8f4e : 80 80 80 8d 80 b0 d6 89  dc
8f56 : 49 03 08 20 5d 02 05 12  45
8f5e : 13 05 0e 04 05 20 02 05  5b
8f66 : 09 0c 09 07 05 0e 04 05  73
8f6e : 13 20 53 03 08 12 09 06  08
8f76 : 14 13 14 5d 03 0b 88 80  70

8f7e : af b1 d6 23 20 09 0e 20  e3
8f86 : 45 12 0c 05 04 09 07 15  47
8f8e : 0e 07 20 49 08 12 05 13  9c
8f96 : 20 53 03 08 12 05 09 02  93
8f9e : 05 0e 13 80 23 20 07 05  d9
8fa6 : 0d 5b 5e 20 14 05 0c 05  a0
8fae : 06 0f 0e 09 13 03 08 05  54
8fb6 : 12 20 41 02 13 10 12 01  65
8fbe : 03 08 05 80 af b2 d6 23  49
8fc6 : 20 0d 09 14 20 44 01 0e  76
8fce : 0b 20 1a 15 12 5d 03 0b  41
8fd6 : 80 23 20 01 0e 13 20 49  7d
8fde : 12 12 0c 5b 15 06 05 12  22
8fe6 : 80 af b3 d6 89 0d 09 14  53
8fee : 20 04 05 12 20 42 09 14  f4
8ff6 : 14 05 20 15 0d 88 80 af  ae
8ffe : b4 d6 23 20 4b 05 0e 0e  1b
9006 : 14 0e 09 13 0e 01 08 0d  e9
900e : 05 80 23 20 42 05 08 01  8f
9016 : 0e 04 0c 15 0e 07 20 17  94
901e : 09 05 20 02 05 13 10 12  40
9026 : 0f 03 08 05 0e 80 23 20  0b
902e : 1d 02 05 12 10 12 5d 06  e3

9036 : 15 0e 07 80 af b5 d6 23  6e
903e : 20 45 12 0c 05 04 09 07  aa
9046 : 15 0e 07 80 23 20 52 5d  6b
904e : 03 0b 07 01 02 05 80 23  49
9056 : 20 56 05 12 02 0c 05 09  cc
905e : 02 80 af b6 d6 4d 09 14  87
9066 : 20 06 12 05 15 0e 04 0c  99
906e : 09 03 08 05 0e 20 47 12  bf
9076 : 5d 5e 05 0e 80 80 80 24  5c
907e : 24 24 24 24 24 24 24 24  7e
9086 : 24 24 24 24 24 24 24 24  86
908e : 24 24 24 24 24 24 24 24  8e
9096 : 24 24 24 24 24 24 24 24  96
909e : 24 24 24 24 24 24 24 24  9e
90a6 : 80 44 01 14 15 0d 71 71  6e
90ae : 71 71 71 71 71 71 71 71  ae
90b6 : 71 71 71 71 55 0e 14 05  8a
90be : 12 13 03 08 12 09 06 14  c5
90c6 : 80 af ff ff 00 00 00 00  1e
```

Listing 1. Der Text für den Begleitnotiz-Zettel. Bitte mit dem MSE (Seite 159) eingeben.



Centronics-Treiber für Giga-Publish

Konnten Sie bisher Drucker, Interface und »Giga-Publish« nicht zu einer befriedigenden Zusammenarbeit überreden? Mit dem neuen Software-Treiber und einem Centronics/User-Port-Kabel erzielen Sie eine fantastische Druckqualität.

Das hervorragende DTP-Programm »Giga-Publish« aus dem Sonderheft 39 liefert Druckergebnisse, die sich sehen lassen können. Beispiele dafür finden Sie in unserem Workshop ab Seite 30. Besitzer eines Druckers mit Centronics-Schnittstelle und einem Kabel zum User-Port des C64 hatten jedoch bisher wenig Gelegenheit, vergleichbare Ergebnisse zu erzielen.

Nur wer zusätzlich über ein Betriebssystem mit integriertem Centronics-Treiber verfügte, konnte »Giga-Publish« sinnvoll einsetzen.

Die Installation eines Software-Treibers, beispielsweise die Eyssele-Schnittstelle aus dem Sonderheft 32, ist nicht ohne weiteres möglich, da die selbststartenden Giga-Publish-Ladeprogramme alle geänderten Kernel-Vektoren auf die ursprünglichen Werte zurücksetzen.

Dieses Problem wird umgangen, wenn ein Programmteil modifiziert wird, das nach dem Start automatisch nachgeladen wird. Um den Aufwand des Abtippens gering zu halten, bietet sich der Programmteil »gpB« an. Er ist nicht sehr lang und bietet von seiner Lage im Speicher genügend Platz für Erweiterungen.

Bevor Sie das Listing 1 mit dem MSE (Seite 159) eingeben und auf der Giga-Publish-Diskette speichern, sollten Sie sicherheitshalber den alten Programmteil »gpB« umbenennen. Legen Sie dazu die Diskette ins Laufwerk und geben folgende Befehle im Direktmodus ein:

```
open1,8,15
print#1,"r:gpB.alt=gpB"
close1
```

Neben dem neuen Software-Treiber benötigen Sie nur noch ein Kabel, das die Verbindung zwischen der Centronics-Schnittstelle des Druckers und dem User-Port des C64 herstellt. Dieses Kabel gibt es im Handel für zirka 30 Mark, läßt sich aber mit etwas Geschick auch selber löten (Materialkosten zirka 15 Mark). Die Anschlußbelegung finden Sie in Tabelle 1.

Bevor Sie das Kabel einstecken, sollten Sie beide Geräte unbedingt ausschalten. Ist die Verbindung hergestellt, laden Sie Giga-Publish wie gewohnt.

Führt der Drucker kein Line-Feed aus, stellen Sie den entsprechenden DIP-Schalter des Druckers auf Auto-LF on. Angaben dazu finden Sie in Ihrem Druckerhandbuch. Eine weitere Möglichkeit besteht auch darin, in der Druckeranpassung die Zeilenvorschub-Sequenz (13) durch eine zusätzliche 10 zu ergänzen.

User-Port		Centronics
A	---GND---	16
B	---Flag2---	10/11
C	---D0----	2
D	---D1----	3
E	---D2----	4
F	---D3----	5
H	---D4----	6
J	---D5----	7
K	---D6----	8
L	---D7----	9
M	--Strobe---	1

Tabelle 1. Anschlußbelegung für ein User-Port/Centronics-Kabel als Verbindung zwischen Drucker und C64

Ein weiterer Vorteil der Verwendung eines User-Port-Kabels besteht in der erhöhten Druckgeschwindigkeit, da die seriell/parallele Wandlung entfällt (je Zeile fallen zum Teil über 5,5 KByte Daten an).

Da »Giga-Publish« nach unseren bisherigen Kenntnissen nicht in jeder Drucker/Interface-Kombination befriedigende Ergebnisse liefert, lohnt sich in solchen Problemfällen ebenfalls die Verwendung des neuen Treibers mit einem User-Port-Kabel.

Die Druckqualität, die sich beim Testen mit einem Star LC-10 unter dieser Kombination ergab, überzeugte einmal mehr von der Qualität dieses DTP-Programms.

(Stefan Seidler/ef)

```
name : gpB c500 c5e8
-----------------------------------
c500 : 20 44 e5 a2 0a a0 0c 20 26
c508 : 0c e5 a9 31 a0 c5 20 1e 8c
c510 : ab a0 53 b9 94 c5 99 3c 6e
c518 : 03 88 10 f7 a9 60 a0 03 88
c520 : 8d 20 03 8c 21 03 a9 3c 59
c528 : a0 03 8d 26 03 8c 27 03 a9
c530 : 60 05 0e c7 49 47 41 2d be
c538 : d0 55 42 4c 49 53 48 20 5e
c540 : d6 31 2e 30 0d 0d 0d 0d c8
c548 : 20 20 20 20 20 57 52 49 1d
c550 : 54 54 45 4e 20 49 4e 20 af
c558 : 31 39 38 39 20 42 59 20 15
c560 : c4 49 45 54 45 52 20 c2 92
c568 : 41 59 45 52 0d 0d 0d 0d 79
c570 : 20 20 20 20 20 20 4d 41 67
c578 : 4e 59 20 54 48 41 4e 4b 64
c580 : 53 20 54 4f 20 d4 48 4f 4b
c588 : 4d 41 53 20 cd 41 4e 47 fd
c590 : 4f 4c 44 00 48 a5 9a c9 c6
c598 : 04 f0 03 4c cd f1 ad 0d 9c
c5a0 : dd 29 10 f0 f9 68 8d 01 4f
c5a8 : dd ad 00 dd 29 fb 8d 00 c0
c5b0 : dd 09 04 8d 00 dd 18 60 d5
c5b8 : 20 0f f3 f0 03 4c 01 f7 01
c5c0 : 20 1f f3 a5 ba c9 04 f0 0d
c5c8 : 03 4c 5b f2 a9 ff 8d 03 fd
c5d0 : dd ad 02 dd 09 04 8d 02 ab
c5d8 : dd ad 0d dd a9 00 20 4e 43
c5e0 : 03 a9 04 85 9a 18 60 46 e2
```

Listing 1. Der Programmteil »gpB« mit zusätzlicher Centronics-Schnittstelle. Geben Sie das Listing bitte mit dem MSE (Seite 159) ein.



Der Weg

Die fantastischen Fähigkeiten des C64 zur Klangerzeugung sind allgemein bekannt. Doch wissen Sie, wie man dem Computer die tollsten Töne und Musikstücke entlockt? Lernen Sie das »Sound-Wunder« in der folgenden Einführung gründlich kennen, und gestalten Sie Ihre Hits selbst.

Zur Erzeugung von Geräuschen und Musik ist der C64 mit einem leistungsfähigen Baustein ausgestattet. Er trägt die Bezeichnung 6581 und soll hier im folgenden SID genannt werden. SID steht für »Sound Interface Device«, was man mit »Klang-Schnittstellen-Baustein« übersetzen könnte. Der SID ist eigentlich ein kleiner Synthesizer, der dreistimmige Melodien spielen oder drei unabhängige Geräusche gleichzeitig erzeugen kann oder auch eine Kombination von beiden, zum Beispiel eine zweistimmige Melodie oder ein Geräusch. Wie man ihn dafür programmiert, soll hier gezeigt werden. Da das Standard-Basic des C64 keine speziellen Befehle zu diesem Zweck vorsieht, muß man sich näher mit dem inneren Aufbau des SID befassen, um ihn dann mit PEEK- und POKE-Befehlen zu steuern. Dieser gezwungenermaßen etwas unelegante Programmierstil hat aber wenigstens einen Vorteil für denjenigen, der in Maschinensprache programmieren kann oder es lernen will. Er kann nämlich die PEEK- und POKE-Befehle direkt in die Assemblersprache übernehmen. Stürzen wir uns also gleich mittenhinein in die SID-Programmierung (in Basic).

Beim SID wird ein Klang durch folgende Parameter (= Steuergrößen) beeinflußt:

1. Lautstärke
2. Hüllkurve — Sie steuert den zeitlichen Lautstärkenverlauf zum Beispiel eines ausklingenden Tones.
3. Kurvenform — Sie ist für den Klangcharakter des Tones verantwortlich.
4. Frequenz — Sie entspricht der Tonhöhe.

Mit Einzelheiten und mit weiteren Parametern zur Klangsteuerung werden wir uns gleich befassen. Zunächst wollen wir aber einmal einen Ton erzeugen, zum Beispiel um zu hören, ob unser Monitor oder Fernseher, der die Töne wiedergeben muß, richtig eingestellt ist (Perfektionisten schließen den C64 über die Audio/Video-Buchse und ein normales DIN-Überspielkabel an die HiFi-Anlage an). Folgende Pokes helfen bei dieser Einstellung:

Der erste Ton

POKE 54296,15	stellt den SID auf maximale Lautstärke
POKE 54278,240	wählt eine einfache Hüllkurve.
POKE 54273,67	stellt eine Frequenz ein (zirka 1000 Hz).
POKE 54276,17	wählt eine sogenannte Dreieckskurve und schaltet zugleich den Ton ein (muß immer als Letztes geschehen!).

Jetzt müßte ein Ton hörbar sein, der ähnlich wie bei einem Fernseh-Testbild klingt.

POKE 54276,16 — schaltet den Ton wieder ab.

Der gleiche Ton mit schärferem Klang gefällig?

POKE 54276,33 — wählt eine »Sägezahnkurve«. Diese klingt heller und schärfer als das Dreieck.

Doch anstatt mit geheimnisvollen POKEs zu arbeiten, sollten wir uns doch besser systematisch mit dem SID befassen. Wer aber nur schnell einen Klangeffekt für ein eigenes Programm benötigt und wen die Einzelheiten des SID nicht so sehr interessieren, der kann den systematischen Teil überspringen und gleich bei »Klangeffekte zum Abtippen« weiterlesen.

Unter einem Register versteht man in der Computertechnik einen Speicherplatz, der mit einer besonderen Funktion gekoppelt ist. Diese Speicherplätze sind also nicht dazu da, um Daten darin abzulegen, sondern um eine Funktion auszulösen oder um Informationen über den Zustand eines Bausteins zu bekommen. Man unterscheidet demnach Schreibregister und Leseregister.

zum richtigen Ton

Der SID verfügt insgesamt über 25 Schreib- und Leseregister. Auf Bild 1 sind diese in grafischer Form dargestellt. Der SID hat die Basisadresse:
S = 54272 (dezimal) oder $D400 (hexadezimal)

Unter dieser und den 28 folgenden Adressen können die Register des SID angesprochen werden. Wir werden in Zukunft Registeradressen wie in Bild 1 immer in der Form S+n (n = 0 bis 28) angeben, weil diese Schreibweise prägnanter als eine fünfstellige Zahl ist. Es ist empfehlenswert, sich auch in Programmen an diese Vereinbarung zu halten.

Das Registerschema gliedert sich in drei Blöcke:
Der erste Block ist in Wirklichkeit dreimal vorhanden, für jede Stimme einmal. Die sieben Register dieser Blöcke haben also für die drei Stimmen unterschiedliche Adressen, wie links im Schema auch angegeben ist.

Der zweite Block (S+21 bis S+24) dient hauptsächlich zur zusätzlichen Klangbeeinflussung durch einen Filter. Den Filter werden wir aber erst später behandeln. Aus diesem Block interessiert zunächst nur die rechte Hälfte des Registers S+24, das für die Lautstärke zuständig ist.

Der dritte Block (S+25 bis S+28) besteht aus vier sogenannten »Nur-Lese-Registern«. Aus diesen Registern kann nur gelesen werden, Schreibzugriffe bleiben wirkungslos. Auch diese Register, die Spezialeffekten dienen, interessieren uns zunächst noch nicht.

Ein Register besteht, wie jeder andere Speicherplatz beim C64 auch, aus einem Byte, beziehungsweise 8 Bit. Man sieht, daß einige Register noch in Felder unterteilt sind. Bei diesen Registern haben einzelne Bits oder Bitgruppen unterschiedliche Bedeutung. Die schraffierten Bereiche kennzeichnen Bits, die keine Funktion im SID haben. Wir werden bald sehen, wie man einzelne Bits innerhalb eines Byte gezielt ansprechen kann. Nun zu den Registern im einzelnen: Es werden beim ersten Block stellvertretend die Register der Stimme 1 (S+0 bis S+6) beschrieben. Die Register für Stimme 2 (S+7 bis S+13) und Stimme 3 (S+14 bis S+20) sind in ihrer Funktion identisch.

Ab hier ist es praktisch, wenn man bei der Lektüre das kleine Programm aus Listing 1 im Computer hat, denn dann kann man die Wirkung der Parameter in den SID-Registern gleich ausprobieren. Die Parameter stehen gut les- und editierbar in den DATA-Zeilen. Das Programm erzeugt nach dem Starten einen Ton bei einem beliebigen Tastendruck. Tasten mit Auto-Repeat-Funktion, wie zum Beispiel die Space-Taste, erzeugen einen Dauerton. Abgebrochen wird das Programm mit der RUN/STOP-Taste. Der letzte Parameter steuert übrigens die Tonlänge durch eine einfache Verzögerungsschleife.

Frequenz S+0 und S+1

Die Frequenz kann beim SID auf 16 Bit genau angegeben werden. Eine 16-Bit-Zahl kann Werte zwischen 0 und 65535 annehmen. Dieser Wert entspricht allerdings nicht der Frequenz in Hz (Hertz = Schwingungen pro Sekunde). Der SID-Wert F zu einer gegebenen Frequenz in Hz errechnet sich nach:

```
F = 17.0284 * Frequenz
```

Der SID-Wert F zum sogenannten Kammerton a mit 440 Hz beträgt also (ganzzahlig gerundet):

```
F = 17.0284 * 440 ≈ 7492
```

Die höchste vom SID erzeugbare Frequenz beträgt dann (gerundet):

```
65535 / 17.0284 ≈ 3849 (Hz)
```

Zum Experimentieren mit Klangeffekten interessiert uns die genaue Frequenz eigentlich gar nicht, für korrekt gestimmte Tonleitern müssen wir sie dagegen kennen. Zunächst wollen wir aber erfahren, wie man den SID mit dem Wert F (Frequenz) programmiert. Diese im Dezimalsystem maximal fünfstellige Zahl wird im Binärsystem durch 16 Bit dargestellt. Da es sich beim C64 um einen 8-Bit-Mikrocomputer handelt, müssen wir diesen Wert in zwei 8-Bit-Hälften, das sogenannte niederwertige und höherwertige Byte, kurz Low-Byte und High-Byte zerlegen. Hier zwei »Rezepte«:

1. Methode (Standard):

```
HI = INT(F/256)
LO = F-256*HI
```

Das ist nichts anderes als eine Division durch 256 mit Rest. HI ist dabei der Quotient und LO der Divisions-

Die Register des SID Basisadresse des SID: S = 54272

Adressen: Stimme 1	Stimme 2	Stimme 3	Bitnummern 7	6	5	4	3	2	1	0
S+0	S+7	S+14	Frequenz-low							
S+1	S+8	S+15	Frequenz-high							
S+2	S+9	S+16	Pulsweite-low							
S+3	S+10	S+17					Pulsweite-high (4 Bit)			
S+4	S+11	S+18					Test	Ringmod	Sync	Gate
S+5	S+12	S+19	Attack				Decay			
S+6	S+13	S+20	Sustain				Release			
		S+21						Filterfrequenz-low		
		S+22	Filterfrequenz-high							
		S+23	Resonanz				Filter Ex	Filter 3	Filter 2	Filter 1
		S+24	S3 Aus	Hoch	Band	Tief	Lautstaerke			
		S+25	Potentiometer X							
		S+26	Potentiometer Y							
		S+27	Oszillator 3							
		S+28	Huellkurve 3							

Bild 1. Alle Register des Sound-Chip auf einen Blick

rest. Die Werte LO und HI sind beides Byte-Werte und liegen damit im Bereich 0 bis 255. Im Fall F=7492 (entsprechend 440 Hz) ergibt sich zum Beispiel:

```
HI = 29 und LO = 68
```

2. Methode (mit Einschränkungen, aber schneller):

```
HI = F/256
LO = F AND 255
```

Die INT-Funktion zur Berechnung von HI wurde hier gespart. HI kann hier noch Nachkommastellen haben; diese werden aber später von dem noch folgenden POKE-Befehl abgeschnitten. Die Berechnung von LO funktioniert hier nur bei F-Werten im Bereich 0 bis 32767. Die zweite Methode ist nur dann zu empfehlen, wenn es auf Geschwindigkeit ankommt.

Mit den Werten LO und HI müssen wir dann die beiden Register S+0 und S+1 besetzen:

```
POKE S+0,LO
POKE S+1,HI
```

Man kann die Wirkungsweise des High- und Low-Bytes auch als Grob- und Feineinstellung auffassen. Oft genügt für einen Klang eine grobe Frequenzsteuerung. Man braucht dann nur das High-Byte zu berücksichtigen und kann das Low-Byte ein für allemal zum Beispiel auf 0 setzen.

Pulsweite S+2 und S+3

Der Parameter »Pulsweite« ist nur wirksam, wenn als Kurvenform das Rechteck gewählt wurde. Die Kurvenformen sind in Bild 1 bei Register S+4 grafisch dargestellt und werden im nächsten Abschnitt besprochen. Das Rechteck ist eine Kurvenform, die nur zwischen zwei Werten hin- und herspringt. Ist der obere Wert genauso lang wie der untere, so spricht man von einer symmetrischen Rechteckkurve. Das Verhältnis zwischen der Länge des oberen und des unteren Wertes kann mit dem Parameter »Pulsweite«, im folgenden P genannt, gesteuert werden. P kann Werte von 0 bis 4095 annehmen und wirkt sich auf die Klangfarbe des Tones aus. Das symmetrische Rechteck, das man mit P = 2048 erhält, klingt verhältnismäßig hohl und wird als typischer Rechteckklang bezeichnet. Entfernt man sich mit P von 2048 in Richtung 0 oder 4095, so wird der Klang zunehmend heller und später schnarrend oder zirpend. Maßgeblich ist hierbei nur der Abstand von P zum Mittelwert 2048. So klingt zum Beispiel P = 2048+500 genauso wie P = 2048-500. Bei P=0 und P=4095 wird kein Ton mehr erzeugt.

P ist eine 12-Bit-Größe und muß wie F in ein Low- und ein High-Byte zerlegt werden. Beim High-Byte können dabei nur die unteren vier Bit gesetzt sein. Zu diesem Zweck kann man ohne Einschränkungen die schon beschriebene Methode 2 anwenden:

```
HI = P/256
LO = P AND 255
POKE S+2,LO
POKE S+3,HI
```

Steuerregister S+4

Dieses Register ist für mehrere Funktionen gleichzeitig zuständig:
- Die Wahl der Kurvenform
- Ein- und Ausschalten des Tones
- Spezialeffekte Ringmodulation und Synchronisation
- Reset der Stimme

Zunächst einmal eine Tabelle mit den Funktionen im einzelnen:

Bit	Dezimal-wert (POKE...)	Funktion
0	1	GATE schaltet Ton ein und aus
1	2	SYNC Synchronisation (Spezialeffekt)
2	4	RING Ringmodulation (Spezialeffekt)
3	8	TEST Reset
4	16	wählt Dreieckskurve
5	32	wählt Sägezahnkurve
6	64	wählt Rechteckkurve
7	128	wählt Rauschen

Mit einem POKE an die Adresse S+4 werden immer alle acht Bit gleichzeitig beeinflußt. Einen Befehl zum Setzen oder Löschen einzelner Bits gibt es nicht. Man muß sich daher über die gewünschten Werte aller acht Bits im klaren sein, auch wenn man nur ein Bit verändern will. Um den richtigen POKE-Wert zu erhalten, müssen die Wertigkeiten der Bits, die man setzen will, addiert werden. Die folgenden drei Beispiele sollen zur Veranschaulichung dienen:

1. Rechteck wählen und Ton einschalten
   Bits: 6 und 0
   = Byte-Wert: 2↑6 + 1 = 65
   POKE S+4,65
2. Ton abschalten, Rechteck gewählt lassen
   Bits: 6
   Byte-Wert: 2↑6 = 64
   POKE S+4,64

Anmerkung: Beim Abschalten eines Tons sollte man immer die zuletzt gewählte Kurvenform gewählt lassen, damit der Ton ausklingen kann. Mit POKE S+4,0 (alle Bits rücksetzen) wird der Ton abrupt abgebrochen.

3. Dreieck mit Ringmodulation wählen, Ton einschalten
   Bits: 4, 2 und 0
   Byte-Wert: 2↑4 + 2↑2 + 2↑0 = 16 + 4 + 1 = 21
   POKE S+4,21

Die Kurvenform

Sie bestimmt die Klangfarbe des Tones. Am vielseitigsten ist das schon besprochene Rechteck, weil man es durch die Pulsweite reichhaltig gestalten kann. Der Sägezahn klingt noch etwas heller und strahlender als das Rechteck. Er eignet sich besonders gut zur Imitation mancher Instrumentenklänge wie Streicher und Blechbläser. Das Dreieck klingt dagegen weich und dumpf und ist bei tiefen Tönen leider leise. Der Klang ist aber bei hohen Tönen sehr angenehm. Rauschen eignet sich für Effekte wie Wind, Düsenlärm, Schüsse, Explosionen und Schlagzeugklänge. Das Rauschen hat zwar keine feste Tonhöhe, doch sein Klangcharakter wird durch den Frequenzparameter entscheidend beeinflußt.

Indem man zwei Kurvenform-Bits gleichzeitig setzt, kann man durch Kombination mehrerer Kurvenformen weitere Klänge erzeugen. Dabei werden die Einzelklänge nicht etwa einfach gemischt, sondern andere Kurvenformen erzeugt. Rauschen läßt sich allerdings nicht mit einer anderen Kurvenform kombinieren. Auch die Kombination von drei Kurvenformen ist unbrauchbar. Sie liefert einen leisen, fast im Rauschen untergehenden Klang. Es bleiben also drei Kombinationen, die sehr interessant klingen:

Wellenform	POKE-Wert (An/Aus)
Rechteck - Sägezahn	97/96
Rechteck - Dreieck	81/80
Sägezahn - Dreieck	49/48

Der Klangcharakter variiert stark von tiefen zu hohen Tönen. Die letzte Kombination liefert nur bei sehr tiefen Tönen gute Resultate. Der Klang der ersten beiden Kombinationen hängt natürlich auch von der Pulsweite P ab.

Die Spezialeffekte

Sie sollen hier nur am Rande erwähnt werden. Wird das SYNC-Bit für Stimme 1 gesetzt (Bit 1 in Register S+4), so kann Stimme 1 nicht mehr frei schwingen, sondern wird von Stimme 3 mit beeinflußt, man sagt hier »synchronisiert«. Auch das Ring-Bit bewirkt, daß Stimme 3 die Stimme 1 beeinflußt. Diese sogenannte Ringmodulation wirkt allerdings nur auf die Dreieckskurve. Der Effekt ist daher nur hörbar, wenn das Ring-Bit (Bit 2) zusammen mit Bit 4 für Dreieck gesetzt wird. Beide Effekte liefern ähnliche Resultate. Es lassen sich unter anderem metallische und glockenähnliche Klänge erzeugen. Die Stimme 3 braucht dabei nicht über ihr GATE-Bit eingeschaltet werden. Maßgeblich ist nur die Frequenz von Stimme 3 (Register S+14 und S+15).

Nun besitzen natürlich auch Stimme 2 und 3 je ein SYNC- und ein RING-Bit. Die drei Stimmen steuern sich dabei nach dem Schema:

Stimme 1 ——► Stimme 2
Stimme 2 ——► Stimme 3
Stimme 3 ——► Stimme 1

Das TEST-Bit wird man wahrscheinlich nie benötigen. Es übt eine lokale Reset-Funktion auf die jeweilige Stimme aus. Solange es gesetzt ist, ist nichts hörbar, unabhängig von den anderen Bits. Wenn man allerdings versucht, Rauschen mit einer anderen Kurvenform zu kombinieren, kann es passieren, daß die betroffene Stimme gewissermaßen »abstürzt« und nichts mehr von sich gibt. Man kann sie dann mit einem gezielten Reset über das TEST-Bit wieder zum Leben erwecken.

Die Hüllkurven S+5 und S+6

Wenn eine Stimme über das GATE-Bit eingeschaltet wird, dann folgt ihr zeitlicher Lautstärkenverlauf einer programmierbaren Hüllkurve. Die Hüllkurve bestimmt unter anderem, ob der Ton hart oder weich einsetzt und ob er schnell oder langsam ausklingt. Der Name kommt von den vier Phasen, die die Hüllkurve durchläuft. Jeder Phase ist dabei ein Parameter zugeordnet.

Attack — Die Attack-Phase wird durch das Setzen des GATE-Bits eingeleitet. Der Pegel steigt dabei von 0 bis Maximum (Lautstärkeregister) an. Die Zeit für diesen Anstieg ist über den Parameter A in 16 nicht-linearen Stufen von 2 ms bis 8 s einstellbar. Eine kurze Attack-Phase bewirkt einen unmittelbaren und harten Toneinsatz wie bei Schlag- oder Zupfinstrumenten. Eine mittlere Attack-Zeit ist typisch für Bläser- und Streicherklänge, und mit einer langen Attack-Zeit kann man einen Ton wie am Mischpult langsam einblenden.

Decay — Nachdem der Maximalwert erreicht ist, fällt der Pegel in der Decay-Phase bis auf den Sustain-Pegel ab. Die Zeit dazu ist mit dem Parameter D in 16 nicht-linearen Stufen von 6 ms bis 24 s einstellbar.

Sustain — nennt man die Phase nach dem Pegelabfall in der Decay-Phase. Der Ton klingt dann so lange auf dem Sustain-Pegel weiter, bis das GATE-Bit zurückgesetzt wird. Der Parameter SU bestimmt hier also keine Zeit, sondern einen Pegel und zwar in 16 linearen Stufen von Null bis Maximum.

Release — Beim Rücksetzen des GATE-Bits wird der Ton nicht einfach abgeschaltet, sondern nimmt in der Release-Phase gleichmäßig vom Sustain-Pegel bis nach Null ab. Die Zeit dazu ist in der gleichen Abstufung wie die Release-Zeit über den Parameter R einstellbar.

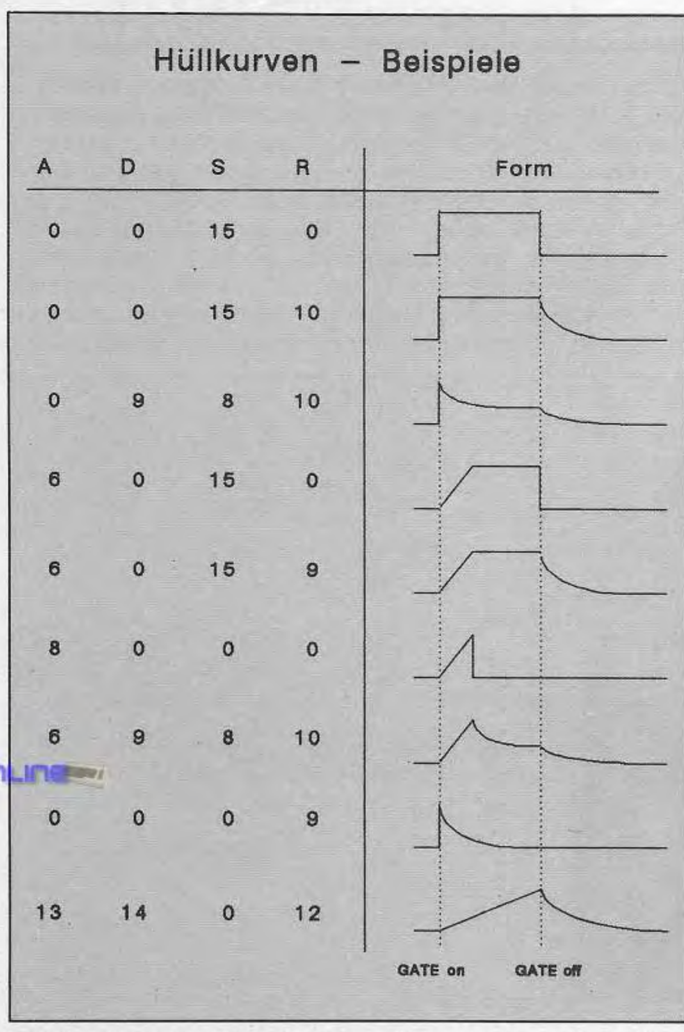


Bild 2. Hier sehen Sie wie sich unterschiedliche Werte für ADSR auf die Müllkurve auswirken

Wenn das GATE-Bit bereits vor Erreichen der Sustain-Phase rückgesetzt wird, dann startet die Release-Phase mit dem aktuellen Pegel der Hüllkurve. Auf diese Weise ergeben sich:

a) Attack-Decay-Release-Zyklen ADR oder gar nur
b) Attack-Release-Zyklen AR

Bei einem Sustainpegel von Null sind Decay und Release funktionell gleichwertig (9. Beispiel in Bild 2). Bei einem maximalen Sustain-Pegel entfällt die Decay-Phase (Attack-Sustain-Release-Zyklus ASR, 1. Beispiel in Bild 2).

Sonderfälle

Die Parameter A, D, SU und R sind 4-Bit-Werte. Jeweils zwei von ihnen werden wie folgt in ein Register gepackt:

```
POKE S+5,16*A+D
POKE S+6,16*SU+R
```

Bild 3 zeigt einige Hüllkurvenbeispiele.

Die bisher erworbenen Kenntnisse befähigen Sie, eindrucksvolle Klänge aus dem SID herauszulocken. Probieren Sie die verschiedenen Möglichkeiten der Klangerzeugung aus. Was uns noch fehlt, ist die weiterführende Beeinflußung der Töne, beispielsweise durch Filter

Der Filter bietet neben der Wahl der Kurvenform eine weitere Möglichkeit zur Klangbeeinflussung. Im Gegensatz zu den vorher beschriebenen Parametern muß man sich aber nicht um den Filter kümmern, da er abschaltbar ist und weil der SID auch ohne Filter reichhaltige Klänge erzeugen

Filter, Register S+21 bis S+24

kann. Um die Wirkungsweise eines Filters zu verstehen, muß man sich einen Klang aus mehreren sogenannten Sinustönen zusammengesetzt denken, dem Grundton und den Obertönen. Sinustöne sind gewissermaßen die nicht mehr weiter zerlegbaren Atome in der Akustik. Ein reiner Sinuston klingt dumpf und ohne charakteristische Färbung. Die vom SID erzeugte Dreiecksschwingung kommt vom Klang her einem Sinuston recht nahe. Die anderen Kurvenformen verdanken ihren helleren Klang einem reichhaltigeren Obertonspektrum (= Folge von Obertönen).

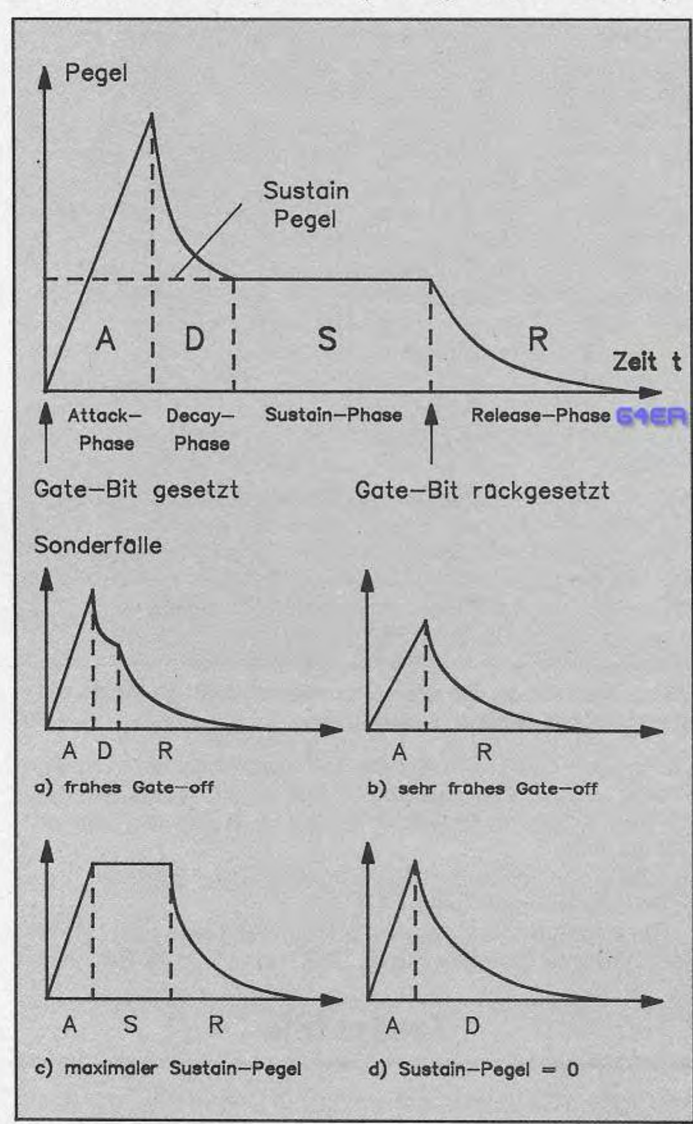


Bild 3. Einige ADSR-Werte und die dazugehörigen Hüllkurven

Und dieses Obertonspektrum kann man mit dem Filter verändern. Der Filter im SID kennt dazu drei Betriebsarten, die über die Bits 4, 5 und 6 in Register S+24 gewählt werden:

Tiefpaß Frequenzen (Obertöne) oberhalb der in den Registern S+21 und S+22 einstellbaren Filterfrequenz werden abgeschwächt, und zwar um so mehr, je höher diese Frequenzen sind. Der Gesamtklang wird dadurch dunkler und weicher.

Hochpaß Es werden die Frequenzen abgeschwächt, die unterhalb der Filterfrequenz liegen. Höhere Frequenzen werden ungehindert durchgelassen. Mit einem Hochpaß kann man den Grundton eines Klanges abschwächen. Seine Gesamtzusammensetzung verschiebt sich dann zugunsten der Obertöne. Der Klang wird dabei dünner und heller.

Bandpaß Dieser Filtermodus schwächt Frequenzen auf beiden Seiten der Filterfrequenz ab. Der Klang wird dabei, wie man fast erwarten kann, etwas dürftig, sofern man nicht maximale Resonanz (siehe weiter unten) einstellt.

Filterfrequenz (Register S+22 und S+23)

Die Filterfrequenz kann auf elf Bit genau eingestellt werden. Dabei kann man aber die drei niederwertigen Bits in Register S+21 unberücksichtigt lassen, da ihr Einfluß praktisch unhörbar ist.

Resonanz und Stimmen-Wahlschalter (Register S+23)

Über den 4-Bit-Parameter Resonanz kann ein gefilterter Klang effektvoller gestaltet werden. Bei großer Resonanz (der Maximalwert ist 15) werden Frequenzanteile in der Gegend der Filterfrequenz verstärkt. Die sonstigen abschwächenden Eigenschaften von Tief- und Hoch- und Bandpaß bleiben dabei erhalten.

Über die Bits 0, 1 und 2 desselben Registers kann man für jede der drei SID-Stimmen unabhängig wählen, ob sie gefiltert oder ungefiltert erklingen soll. Ist zum Beispiel Bit 0 gesetzt, so wird Stimme 1 gefiltert. Das Bit 3, Filter Ex, steuert die Verarbeitung einer von außen zuführbaren Signalquelle, zum Beispiel eines zweiten SID.

Filtermodus und Lautstärke (Register S+24)

Die Lautstärkeneinstellung haben wir schon kennengelernt. Man wird sie meistens auf ihren Maximalwert 15 stellen, weil dann der Rauschabstand und damit die Klangqualität am besten ist. Durch Setzen der Bits 4, 5 und 6 wird die Betriebsart des Filters, Tief-, Band- oder Hochpaß gewählt. Die Betriebsarten sind uneingeschränkt kombinierbar. Mit Bit 7 (S3 Aus) kann man die Stimme 3 unhörbar machen. Der Sinn dieser Funktion wird in folgendem Abschnitt klar.

Ein Beispiel zur Filterprogrammierung:

Stimme 1 soll mit maximaler Resonanz durch den Tiefpaßfilter geschickt werden:

```
FF  = 50      Filterfrequenz (nur High-Byte)
FR  = 241     (=15*16 für Resonanz +1 für Bit 1)
ML  = 31      (=16 für Bit 4 + 15 für Lautstärke)
POKE S+22,FF
POKE S+23,FR
POKE S+24,ML
```

Die Leseregister S+25 bis S+28

Die Register S+25 und S+26 haben mit der Klangprogrammierung nichts zu tun. Über sie können die Werte zweier an Joystick-Ports angeschlossener Potentiometer (Paddles)

Register	A S+5	D S+5	SU S+6	R S+6	C S+4	P S+2 S+3	F S S+1	FF S+22	FR S+23	ML S+24	M
Glöckchen	0	10	0	10	16	x	40000	x	0	15	100
Oboe	8	7	10	8	64	250	7500	x	0	15	500
Fagott	8	7	10	8	64	250	2500	x	0	15	750
Zungenpfeife	8	0	15	10	48	x	400	x	0	15	1000
Banjo	0	8	0	8	32	x	7500	50	241	111	30
Stahl	0	0	15	12	96	2044	30000	x	0	15	100
Feder	0	8	0	9	32	x	750	x	0	15	35
Preßlufthammer	0	0	15	10	80	2100	200	x	0	15	2000
Schuß	0	8	0	10	128	x	10000	x	0	15	50
Starkstrom	0	0	15	0	128	x	100	x	0	15	2000
Düsenflugzeug	0	0	15	13	128	x	3000	50	241	31	3000
Rakete	0	0	15	15	128	x	1000	10	241	31	3000

x = don't care (Parameter muß nicht eingestellt werden)

Tabelle 1. Einige Beispiel-Effekte für Listing 1. Die Parameter müssen in den Zeilen 320 bis 390 verändert werden.

abgefragt werden. Interessant sind die Register S+27 und S+28:

Aus S+27 kann man den Signalverlauf von Stimme 3 in Form von Byte-Werten lesen. Mit folgendem kleinen Programm kann man diesen Signalverlauf sogar sichtbar machen:

```
10 S=54272
20 POKE S+14,10        :REM F LOW
30 POKE S+15,0         :REM F HIGH
40 POKE S+18,16        :REM DREIECK
50 PRINT TAB(PEEK(S+27)/7);" *":GOTO 50
```

Hier sollte man einmal ein wenig mit den Parametern in Zeile 20–40 experimentieren.

Fest eingestellt: Register	FF S+22	FR S+23	ML S+24
	x	0	15

Register	A S+5	D S+5	SU S+6	R S+6	C S+4	P S+2 S+3	F S S+1	G	N	M
Telefon	0	10	0	10	16	x	16000	1.33	2	25
Laserkanone	0	0	15	0	64	1000	30000	0.85	10	1
Take-Off	0	0	15	15	128	x	500	1.004	1000	1
Turbine	0	0	15	15	96	2044	20000	1.001	460	1
Trommelwirbel	0	5	2	9	128	x	20000	1	2	30
Maschinengewehr	0	5	2	9	128	x	12000	0.7	3	30

x = don't care (Parameter muß nicht eingestellt werden)

Tabelle 2. Einige Beispiel-Effekte für Listing 2. Die Parameter müssen in den Zeilen 420 bis 480 verändert werden.

Auf die gleiche Weise kann man aus Register S+28 den Hüllkurvenverlauf von Stimme 3 lesen.

```
100 S=54272
110 POKE S+19,16+11+11     :REM A D
120 POKE S+20,16*8 +11     :REM S R
130 POKE S+18,1            :REM GATE ON
140 FOR I=1 TO 50
150 PRINT TAB(PEEK(S+28)/7);" *"
160 NEXT I
170 POKE S+18,0            :REM    GATE OFF
180 FOR I=1 TO 50
190 PRINT TAB(PEEK(S+28)/7);" *"
200 NEXT I
```

Diese Werteverläufe sind besonders zum Modulieren anderer Stimmen geeignet. Normalerweise möchte man Stimme 3 dann nicht hören, wenn man ihren Signalverlauf oder ihre Hüllkurve anderweitig verwendet. Man kann sie dann, wie schon erwähnt, über Bit 7 in Register S+24 ausschalten.

Klangeffekte zum Abtippen

Nach diesem systematischen Teil folgen noch Einstellungen. Tabelle 1 enthält einige Parametersätze für Klänge, die der SID ohne großen Programmieraufwand erzeugen kann. Die Klangbezeichnungen wollen die Effekte nur subjektiv beschreiben und sind natürlich nicht zu wörtlich zu nehmen. Man muß nun lediglich die Werte einer Zeile in die, in der Kopfzeile angegebenen SID-Register schreiben, das GATE-Bit setzen und nach einiger Zeit zurücksetzen. Der Parameter M ist übrigens kein SID-Parameter, sondern soll eine Verzögerungsschleife steuern, die die Zeit zwischen GATE ON und GATE OFF bestimmt. Parameter M bezieht sich auf das Programm in Listing 1, das beim Experimentieren Hilfestellung leisten soll. Im DATA-Teil ab Zeile 320 sind die Parameter aus Tabelle 1 einzusetzen und zwar genau in der gleichen Reihenfolge. Das Programm belegt nach dem Start den SID mit den Parametern aus den DATA-Zeilen und wartet auf einen beliebigen Tastendruck, der dann den Klangeffekt auslöst. Man versuche es auch einmal mit den Tasten, die eine Auto-Repeat-Funktion haben, wie zum Beispiel die Space-Taste.

Das Programm aus Listing 2 ist ganz ähnlich aufgebaut, kann aber ein viel größeres Spektrum von Effekten dadurch realisieren, daß es die Frequenz von Stimme 1 dynamisch

Register	A	D	SU	R	C	P	F	G	N	M	A3 S+19	B3 S+19	F3 S+20	R3 S+20	C3 S+18	P3 S+16 S+17	F3 S+14 S+15	Q
Vogelgezwitscher	0	8	0	8	16	x	40000	500	8	10	0	8	0	0	x	x	x	28
Bongo	0	7	0	7	16	x	4000	1000	4	1	0	8	0	0	x	x	x	28
E-Baß	0	8	0	9	32	x	750	1000	7	1	0	10	0	0	x	x	x	28
Dampfhammer	0	9	0	11	128	x	5000	200	20	15	0	9	0	9	x	x	x	28
Martinshorn	0	0	15	8	64	1000	7000	720	300	1	x	x	x	x	64	2048	15	27
Sirene	10	13	0	0	64	2048	10000	400	300	1	x	x	x	x	16	x	30	27
Geklimper	0	0	15	0	64	2048	10000	200	400	1	x	x	x	x	128	x	20	27
Grollen	9	10	0	0	32	x	50000	2000	40	10	x	x	x	x	128	x	500	27

Tabelle 3. Einige Beispiel-Effekte für Listing 3. Die Parameter müssen in den Zeilen 610 bis 650 verändert werden.

verändert. Dazu dient die innere Schleife, Zeile 260–300. Dort wird bei jedem Durchlauf die Frequenz F1 mit einem Faktor G multipliziert. Für G>1 steigt die Frequenz schneller an, für G<1 nimmt sie ab und zwar um so schneller, je weiter G von 1 entfernt ist. Die Umrechnung von F1 in Low- und High-Byte geschieht hier nach der schnellen Methode 2. Man beachte, daß damit nur F1-Werte bis 32767 verarbeitet werden können, also nur die Hälfte des vollen Frequenzumfangs des SID. In der äußeren Schleife wird der Wert von F1 auf seinen Ausgangswert F zurückgesetzt. Außerdem steuert die äußere Schleife bei jedem Durchlauf einen ADSR-Hüllkurvenzyklus durch GATE-ON-GATE-OFF. Die Zahl N gibt dabei die Anzahl der inneren Schleifendurchläufe an, die Zahl M die der äußeren Schleifendurchläufe. Beispielparameter zu Listing 2 findet man in Tabelle 2.

Das dritte Programm (Listing 3) verwendet schließlich den Signalverlauf oder die Hüllkurve von Stimme 3, um Stimme 1 zu modulieren. Dies geschieht in der inneren Schleife, Zeile 360–380. Q ist dabei die Adresse eines der Leseregister des SID, also entweder S+27 für den Signalverlauf oder S+28 für den Hüllkurvenverlauf. In der DATA-Zeile 650 ist dazu nur 27 oder 28 anzugeben. Über die Variable G kann die Stärke der Modulation, die sogenannte Modulationstiefe, gesteuert werden. Ein kleinerer Wert von G ergibt hier eine stärkere Modulation. Die äußere Schleife steuert hier ADSR-Hüllkurvenzyklen für Stimme 1 und Stimme 3. Dieses kleine Programm sollte Anlaß zum weiteren Experimentieren sein. Man kann statt der Tonfrequenz zum Beispiel auch einmal versuchen, die Pulsweite oder die Filterfrequenz zu modulieren. Tabelle 3 enthält einige Parametersätze für dieses Programm.

Ausblick: Programmierung von Musikstücken

Dieses Thema wollen wir hier nur einmal streifen. Grundlage hierfür ist die genaue Kenntnis der Tonleiterfrequenzen. Diese müssen aber nicht mühsam aus Tabellen abgetippt, sondern können durch ein kleines Programm selbst berechnet werden. Man muß dazu ein klein wenig Mathematik betreiben:

1) Die Frequenzen zweier Töne im Oktavabstand verhalten sich wie 2:1.
2) Eine Oktave ist durch die Halbtöne in zwölf gleiche Intervalle eingeteilt und zwar nicht linear, sondern exponentiell.
3) Das Frequenzverhältnis H zweier aufeinanderfolgender Halbtöne ist die zwölfte Wurzel aus zwei. In Basic läßt sich das leicht ausrechnen:

   $H = 2^{1/12}$
4) Man bekommt dann alle Halbtöne, indem man die Frequenzen, ausgehend von einem Grundwert, fortlaufend mit H multipliziert.

Im Programm aus Listing 4 macht das eine Schleife in Zeile 140 bis 190. Die Variable FAUS wird mit 110 vorbesetzt. Das entspricht einem »großen A«, dem Ton, der zwei Oktaven unter dem Kammerton, dem bekannten »eingestrichenen a« mit 440 Hz, liegt. In Zeile 170 wird FAUS in den dazugehörenden SID-Wert F umgerechnet. F wird in High- und Low-Byte zerlegt, welche in den Feldern FH und FL gespeichert werden. Dort stehen die Töne als fertige POKE-Werte zum Spielen einer Melodie zur Verfügung.

Das restliche Programm erzeugt aus diesen Werten eine mehr oder weniger zufällige Tonfolge auf dem SID. Ein wenig wird der Zufall aber durch die Werte in den DATA-Zeilen ab Zeile 500 gesteuert. Diese Werte stellen ein Blues-Schema in codierter Form dar. Das Schema selbst steht in den letzten drei Zeilen. Das Programm holt sich aus diesem Schema die erste Zahl. Diese zeigt auf eine der sieben Auswahlmengen in den vorausgehenden Zeilen. Die Auswahlmengen enthalten Tonnummern. Das Programm spielt nun nacheinander acht zufällig ausgewählte Töne aus dieser Menge. Tonwiederholungen sind dabei möglich. Anschließend geht das Programm über den nächsten Zeiger aus dem Schema zu einer neuen Auswahlmenge über. Die links etwas abgesetzten Zahlen dienen dabei nur zur Längenangabe der Auswahlmengen und des Schemas.

Mit dieser einfachen Technik wird der Blues-Charakter deutlich hörbar. (Thomas Krätzig/rs/kn)

```
100 REM---------------------------------     <146>
110 REM EINFACHE KLANGEFFEKTE                <107>
120 REM---------------------------------     <166>
130 S=54272                                  <150>
140 READ A,D,SU,R,C,P,F,M,FF,FR,ML           <187>
150 POKE S+5 ,16*A +D                        <174>
160 POKE S+6 ,16*SU+R                        <037>
170 POKE S+2 ,P AND 255                      <085>
180 POKE S+3 ,P/256                          <133>
190 HI=INT(F/256):LO=F-256*HI                <142>
200 POKE S   ,LO                             <222>
210 POKE S+1 ,HI                             <018>
220 POKE S+22,FF                             <056>
230 POKE S+23,FR                             <154>
240 POKE S+24,ML                             <224>
250 GET A$:IF A$="" THEN 250                 <220>
260 : POKE S+4,C OR 1                        <191>
270 : FOR I=1 TO M:NEXT I                    <161>
280 : POKE S+4,C AND 254                     <193>
290 GOTO 250                                 <052>
300 REM---------------------------------     <092>
310 REM PARAMETER                            <224>
320 DATA 0 ,10, 0,10:REM A D S R             <165>
330 DATA 16          :REM CONTROL-BYTE       <222>
340 DATA 2048        :REM PULSWEITE          <129>
350 DATA 40000       :REM FREQUENZ           <183>
360 DATA 100         :REM VERZOEGERUNG       <216>
370 DATA 50          :REM FILTERFREQUENZ     <033>
380 DATA 0           :REM FILTERRESONANZ     <148>
390 DATA 15          :REM MODUS/LAUT         <209>
```

Listing 1. Einfache Klangeffekte. Hinweise zum Text. Programm bitte mit dem Checksummer (Seite 158) eingeben.

```
100 REM---------------------------------     <146>
110 REM KLANGEFFEKTE MIT                     <098>
120 REM DYNAMISCHER FREQUENZSTEUERUNG        <218>
130 REM---------------------------------     <176>
140 S=54272                                  <160>
150 READ A,D,SU,R,C,P,F,G,N,M                <206>
160 POKE S+5 ,16*A +D                        <184>
170 POKE S+6 ,16*SU+R                        <047>
180 POKE S+2 ,P AND 255                      <095>
190 POKE S+3 ,P/256                          <143>
200 POKE S+23,0       :REM FR                <203>
210 POKE S+24,15      :REM ML                <057>
220 GET A$:IF A$="" THEN 220                 <253>
230 FOR I=1 TO M                             <086>
240 : F1=F                                   <004>
250 : POKE S+4,C OR 1                        <179>
260 : FOR J=1 TO N                           <224>
270 :    POKE S,F1 AND 255                   <039>
280 :    POKE S+1,F1/256                     <178>
290 :    F1=F1*G                             <010>
300 : NEXT J                                 <206>
310 : POKE S+4,C                             <204>
320 NEXT I                                   <150>
330 GOTO 220                                 <044>
400 REM---------------------------------     <192>
410 REM PARAMETER                            <068>
420 DATA 0 ,10, 0,10:REM A D SU R            <088>
430 DATA 16          :REM CONTROL-BYTE C     <092>
440 DATA 2048        :REM PULSWEITE P        <038>
450 DATA 16000       :REM FREQUENZ F         <124>
460 DATA 1.33        :REM FAKTOR G           <117>
470 DATA 2           :REM ANZAHL N           <197>
480 DATA 25          :REM ANZAHL M           <084>
```

Listing 2. Klangeffekte mit Frequenzsteuerung

```
100 REM---------------------------------     <146>
110 REM KLANGEFFEKTE MIT                     <098>
120 REM DYNAMISCHER STEUERUNG                <139>
125 REM DURCH STIMME 3                       <066>
130 REM---------------------------------     <176>
140 S=54272                                  <160>
150 READ A,D,SU,R,C,P,F,G,N,M                <206>
160 POKE S+5 ,16*A +D                        <184>
170 POKE S+6 ,16*SU+R                        <047>
180 POKE S+2 ,P AND 255                      <095>
190 POKE S+3 ,P/256                          <143>
200 POKE S+23,0        :REM FR               <203>
210 POKE S+24,128+15:REM ML (S3 AUS)         <086>
220 READ A3,D3,S3,R3,C3,P3,F3,Q              <190>
230 HI=INT(F3/256):LO=F3-256*HI              <201>
240 POKE S+14,LO                             <197>
250 POKE S+15,HI                             <255>
260 POKE S+16,P3 AND 255                     <111>
270 POKE S+17,P3/256                         <209>
280 POKE S+19,16*A3+D3                       <209>
290 POKE S+20,16*S3+R3                       <204>
300 Q=S+Q                                    <230>
310 F=F/256                                  <091>
320 GET A$:IF A$="" THEN 320                 <131>
330 FOR I=1 TO M                             <188>
340 : POKE S+4 ,C OR 1                       <015>
350 : POKE S+18,C3 OR 1                      <012>
360 : FOR J=1 TO N                           <068>
370 :    POKE S+1,F*(1+PEEK(Q)/G)            <106>
380 : NEXT J                                 <030>
390 : POKE S+4 ,C                            <028>
400 : POKE S+18,C3                           <166>
410 NEXT I                                   <240>
420 GOTO 320                                 <142>
500 REM---------------------------------     <036>
510 REM PARAMETER STIMME 1                   <250>
520 DATA 0 , 8, 0, 8:REM A D SU R            <243>
530 DATA 32           :REM CONTROL-BYTE C    <146>
540 DATA 2048         :REM PULSWEITE P       <140>
550 DATA 40000        :REM FREQUENZ F        <154>
560 DATA 500          :REM FAKTOR G          <239>
570 DATA 8            :REM ANZAHL N          <091>
580 DATA 10           :REM ANZAHL M          <098>
600 REM PARAMETER STIMME 3                   <094>
610 DATA  0, 8, 0, 0:REM A3 D3 S3 R3         <144>
620 DATA 16           :REM CONTROL C3        <233>
630 DATA 2048         :REM PULSWEITE P3      <127>
640 DATA 10           :REM FREQUENZ F3       <063>
650 DATA 28           :REM MOD.-QUELLE Q     <201>
```

Listing 3. Klangeffekte mit Steuerung durch Stimme 3

```
10 REM----------------------------------     <161>
20 REM ZUFALLSTONFOLGE                       <234>
30 REM MIT BLUES-SCHEMA                      <079>
40 REM                                       <102>
50 REM AUSNUETZUNG ALLER DREI STIMMEN        <099>
60 REM ZUR KLANGVERBESSERUNG                 <238>
70 REM                                       <132>
80 REM T. KRAETZIG           MAERZ 86        <245>
90 REM----------------------------------     <241>
100 DIM FL(25) :REM ARRAY F. FREQUENZEN      <058>
101 DIM FH(25)                               <137>
102 DIM A(8,20):REM AUSWAHLMENGEN            <164>
104 DIM S(30)   :REM SCHEMA                  <131>
110 S  =54272   :REM BASISADRESSE            <235>
130 :                                        <106>
140 REM TONLEITER-FREQUENZEN BERECHNEN       <166>
150 FAUS=110:H=2↑(1/12)                      <026>
160 FOR I=0 TO 25                            <152>
170 :  F=INT(FAUS*17.0284+0.5)               <123>
172 :  FH(I)=INT(F/256)                      <213>
174 :  FL(I)=F-256*FH(I)                     <097>
180 :  FAUS=FAUS*H                           <067>
190 NEXT I                                   <018>
200 :                                        <176>
210 REM PARAMETER FESTLEGEN                  <133>
220 PW=2048           :REM PULSWEITE         <137>
230 C =32             :REM KURVENFORM        <135>
240 A=0:D=10:SU=0:R=9                        <192>
250 FOR I=0 TO 14 STEP 7                     <159>
255 : POKE S+I+2,PW AND 255                  <128>
260 : POKE S+I+3,PW/256                      <233>
265 : POKE S+I+5,16*A+D                      <160>
270 : POKE S+I+6,16*SU+R                     <024>
275 NEXT I                                   <105>
280 :                                        <002>
290 REM FILTER AUS UND LAUTSTAERKE MAX.      <155>
300 POKE S+23,0:POKE S+24,15                 <066>
310 :                                        <032>
320 REM AUSW.MENGEN UND SCHEMA EINLESEN      <240>
325 READ I:A(0,0)=I                          <245>
330 FOR K=1 TO I                             <202>
335 :  READ J:A(K,0)=J                       <228>
340 :  FOR L=1 TO J:READ A(K,L):NEXT L       <013>
350 NEXT K                                   <196>
355 READ I:S(0)=I                            <175>
360 FOR K=1 TO I                             <232>
365 :  READ S(K)                             <051>
370 NEXT K                                   <216>
375 :                                        <097>
380 REM ZUFALLSTONFOLGE                      <084>
385 L=0:O=0                                  <239>
390 FOR I=1 TO S(0)                          <039>
395 :  J=S(I)                                <051>
400 :  N=A(J,0)                              <158>
405 :  FOR K=1 TO 8                          <107>
410 :     ZZ=A(J,INT(RND(1)*N+1))            <064>
412 :     POKE S+L  ,FL(ZZ+O)                <166>
414 :     POKE S+L+1,FH(ZZ+O)                <115>
416 :     POKE S+L+4,C OR 1                  <105>
420 :     FOR P=1 TO 40:NEXT                 <035>
425 :     POKE S+L+4,C                       <218>
430 :     FOR P=1 TO 40:NEXT                 <045>
435 :     L=L+7:IF L=21 THEN L=0             <121>
440 :  NEXT K                                <106>
445 NEXT I                                   <019>
450 O=O+1:IF O=4 THEN O=0                    <003>
455 FOR P=1 TO 1150:NEXT                     <183>
460 GOTO 390                                 <038>
500 REM---------------------------------     <036>
510 REM AUSWAHLMENGEN UND SCHEMA             <016>
520 REM---------------------------------     <058>
530 DATA  7                                  <221>
540 DATA  8,  0,4,7,10,12,16,19,22           <243>
550 DATA  8,  0,3,5,9, 12,15,17,21           <238>
560 DATA  7,  2,5,7,11,14,17,19              <126>
570 DATA  6,  0,0,4, 7, 7,10                 <096>
580 DATA  4,  0,3,5,9                        <114>
590 DATA  5,  2,5,7,7,11                     <237>
610 DATA  2,  0,7                            <115>
620 :                                        <088>
630 DATA 24,  7,1,2,1,3,2,1,3                <210>
640 DATA      4,4,5,4,6,5,4,6                <058>
650 DATA      7,1,2,1,3,2,1,3                <117>
```

Listing 4. Programm zum Errechnen einer Zufalls-Blues-Musik. Bitte alle Listings mit dem Checksummer (Seite 158) eingeben.



BASIC FÜR AU

1. Grundlagen der String-Verarbeitung

Durch die geschickte Nutzung von Strings und Variablen ergeben sich in Basic vielfältige Gestaltungsmöglichkeiten. Viele scheuen jedoch dieses Gebiet der Programmierung. Unser Kurs schafft hier Abhilfe.

Wir werden uns intensiv mit der Behandlung von Strings, also Buchstaben und Zeichenketten, beschäftigen. Obwohl wir dabei in kleinen Schritten und mit vielen praktischen Beispielen vorgehen, sollten Sie einige Basic-Grundkenntnisse mitbringen. Für absolute Neulinge empfehlen wir daher unseren Basic-Grundkurs aus dem Sonderheft 40.

Unser String-Kurs ist in sieben Teile gegliedert. Im ersten Abschnitt werden wir zunächst die Grundlagen der Strings besprechen. Sie lernen darin neun Basic-Befehle und sieben numerische Funktionen zur Verarbeitung von Strings kennen.

Teil 2 befaßt sich mit Texteingabe und Textverarbeitung von Strings. Weitere Themen sind das Formatieren von Text und Zahlen (Teil 3), das Erzeugen von Laufschriften (Teil 4), das Sortieren von Zeichen und Wörtern (Teil 5), Suchverfahren mit Hilfe von Stringoperationen (Teil 6) und letztlich die Gestaltung von Benutzermenüs, Tabellen und (Highscore-)Listen (Teil 7). Alle Teile des Kurses bauen dabei kontinuierlich aufeinander auf. Die Kenntnisse, die Sie mit unserem Kurs erwerben, versetzen Sie in die Lage, Ihre eigenen Basic-Programme effektiver zu entwickeln. In vielen Fällen reicht Basic aus, um sinnvolle Anwendungen zu programmieren. Eine Zusammenstellung der verschiedenen Teile finden Sie in einer Kursübersicht auf Seite 52.

Im Prinzip ist der Heimcomputer ein überdimensionaler Taschenrechner. Das Wort »Computer« bedeutet auch nichts anderes als »Rechner«. Es sind jedoch nicht nur Bildschirm, Floppystation oder Tastatur, die den Unterschied machen. Entscheidend ist die Fähigkeit des Heimcomputers, neben Zahlen auch Buchstaben, Texte, Zeichen und Grafiken verarbeiten zu können.

Bereits der Befehl PRINT gibt dem Einsteiger die Möglichkeit, Überschriften und Texte in seine Programme einzubauen. Doch Basic bietet mehr. Texte können innerhalb eines Programmablaufes miteinander verglichen, verändert, bewegt oder sonstwie manipuliert werden. Und das alles mit Strings. Schon wieder so ein Fachwort? Nicht direkt. Das Wort »String« stammt aus dem Englischen und bedeutet soviel wie Kette, Schnur oder Reihe. Man kann sich vorstellen, daß man viele Buchstaben und Zeichen auf einer Schnur oder, wie einzelne Glieder, zu einer Kette zusammenfaßt. Diese Kette ergibt dann ein Wort oder eine bestimmte Zeichenkombination, wie in Bild 1 veranschaulicht wird. Solch eine Zeichenkette ist nichts anderes als ein String.

Da Strings für den Basic-Programmierer ein unverzichtbares Werkzeug sind, haben wir uns vorgenommen, in sieben Lektionen die verschiedenen Anwendungsmöglichkeiten zu beschreiben. Die Stringbefehle sind bei allen Commodore-Computern praktisch gleich. Diese Serie gilt demnach für die Commodore-Computer C 64, C 128, C 16/116, Plus/4 und den betagten VC 20. Eventuelle feine Unterschiede werden natürlich einzeln behandelt.

In einem Programm ist es sehr nützlich, daß der Computer sich bestimmte Zahlen oder Strings merken kann. »Merken« bedeutet, er kann sie im Speicher festhalten und je-

derzeit dort wieder herausholen, vorausgesetzt der Programmierer kennt ihre Namen. Es ist nämlich nicht ohne weiteres möglich, dem Computer zu befehlen: Merke dir die Zahl 7 oder das Wort »Commodore«. Mit dem Begriff »Variable« werden Namen bezeichnet, unter denen sich der Computer Zahlenwerte oder Strings merkt. Das können wir gleich ausprobieren. Geben Sie direkt ein:

```
X=215:Y=0.34 <RETURN>
```

Sie haben nun einer Variablen mit dem Namen X den Wert 215 zugewiesen, einer anderen mit dem Namen Y den Wert 0.34. Diese Werte können jederzeit aus dem Speicher geholt und auf den Bildschirm gebracht werden. Geben Sie wieder direkt über die Tastatur ein:

```
PRINT X:PRINT Y <RETURN>
```

Versuchen Sie dieses einmal mit Strings, zum Beispiel:

```
A=NAME <RETURN>
```

Bereits hier zeigt der Computer die Fehlermeldung ?SYNTAX ERROR. Strings besitzen eine Besonderheit. Sie müssen immer zwischen Anführungszeichen (Gänsefüßchen) stehen. Der Computer erkennt so, wo ein String anfängt und aufhört. Fehlen die Anführungszeichen, weiß er nichts mit der Variablen anzufangen. Versuchen Sie es noch einmal:

```
A="NAME" <RETURN>
```

Der Computer meldet ?TYPE MISMATCH ERROR. Offensichtlich haben wir noch etwas falsch gemacht. Basic kennt zwei Arten von Variablen, numerische Variablen, denen wir Zahlenwerte zuordnen, und String-Variablen, die für Strings gebraucht werden. Eine String-Variable wird mit dem Dollarzeichen ($) gekennzeichnet. Es befindet sich über der Taste <4> auf der Tastatur. Bild 2 veranschaulicht, was zu wem paßt, oder nicht. Nun können wir es nochmals versuchen:

```
A$="NAME" <RETURN>
```

(Der Computer meldet READY.)

```
PRINT A$ <RETURN>
```

Eine Besonderheit ist noch zu beachten. Geben Sie ein:

```
SPRINT$="NAME" <RETURN>
```

Schon wieder ?SYNTAX ERROR. Es ist aber doch alles richtig eingegeben, oder? Nicht ganz. Unsere Variable SPRINT$ enthält den Basic-Befehl PRINT. Das »S« davor wird als falsche Schreibweise angesehen und führt zu einer Fehlermeldung. Basic-Befehle dürfen also nicht in Variablen-Namen enthalten sein. Versuchen Sie es ruhig einmal mit den Variablen CONTAINER$, DEFEKT$, DIMMER$, ENDE$, STIFT$ oder LISTE$. Sie werden immer dasselbe Ergebnis erhalten. Es ist ratsam, auf »schöne« Variablen-Namen zu verzichten, statt dessen eine Kombination eines Buchstaben mit einer Zahl zu wählen, zum Beispiel A3$ oder ZZ$. Voraussetzung ist, der Variablenname beginnt mit einem Buchstaben. 3A$ ist keine zulässige Variable.

Strings wären relativ nutzlos, könnte man sie lediglich irgendwo speichern und wieder aufrufen. Basic stellt neun Befehle und sieben numerische Funktionen bereit, die Strings verändern, verschieben, erweitern und manipulieren. Die einfachste Manipulation ist die Addition von

Wer mit Buchstaben und Zeichenketten mehr machen will als nur Texte schreiben, dem bietet unser Kurs eine Fülle von Anregungen für eigene Programme.

Strings über den PRINT-Befehl. Geben Sie das kleine Listing 1 ein und starten Sie es mit RUN <RETURN>.

In den Zeilen 110 und 115 weisen wir den Variablen A$ und B$ die Strings "HOLZ" und "FEUER" zu. In Zeile 120 lassen wir den Computer einfach beide Strings nebeneinander schreiben, so daß das Wort »HOLZFEUER« entsteht. Es ist egal, ob zwischen A$ und B$ ein Semikolon, ein Zwischenraum oder nichts steht. Dem neuen Wort »HOLZFEUER« können wir auch eine neue Variable C$ zuordnen.

Wir addieren Strings

Erweitern Sie Listing 1 mit folgenden Programmzeilen und starten es wieder mit RUN.

```
125 C$=A$+B$
130 PRINT C$
```

Es erscheint zweimal »HOLZFEUER«. In Zeile 125 haben wir festgelegt, daß die Variable C$ eine zusammengesetzte Zeichenkette aus den bekannten Strings A$ und B$ sein soll. Das neue Wort bleibt so auch später verfügbar. Geben Sie direkt ein:

```
PRINT C$ <RETURN>
```

Nicht ganz so selbstverständlich ist die Addition von Strings, die aus Zahlen bestehen. Geben Sie Listing 2 ein, starten Sie es mit RUN <RETURN>.

Nummer	Inhalt	Seite
1	Grundlagen der String-Verarbeitung	50
2	Texteingabe und Textverarbeitung	56
3	Formatieren von Texten und Zahlen	63
4	Laufschrift und Farbenspiele	67
5	Spielereien mit Wörtern und Texten	71
6	Suchverfahren, Wörterraten	75
7	Menüs, Tabellen, Listen	79

Der Inhalt des Kurses im Überblick

Zeile 130 druckt die Zahl 75 aus. Bemerkenswert ist dabei, daß die Ziffern ohne den sonst üblichen freien Platz für ein eventuelles negatives Vorzeichen gedruckt werden. Eine Eigenschaft, die wir uns später noch zunutze machen werden.

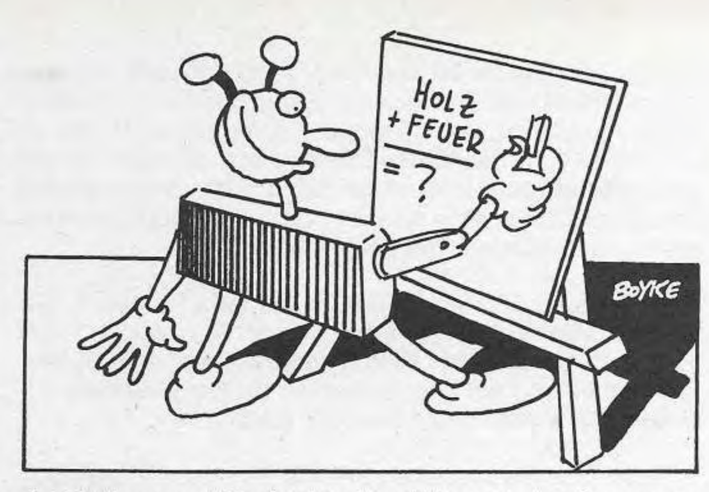
Einen schönen Effekt erhält man durch Darstellung von Steuerzeichen wie »CURSOR-LINKS« oder »REVERS-ON« als Strings. Geben Sie Listing 3 ein und starten es mit RUN. Zeile 140 druckt das Wort »FEUERHOLZ« aus, wobei der Wortteil »FEUER« in reverser Darstellung erscheint.

Auch die Grafikzeichen, auf die mit der <SHIFT>- oder <COMMODORE>-Taste umgeschaltet werden, können als Strings addiert werden (Listing 4). Durch Variablen, die aus Zeichen und Cursorbewegungen bestehen, können Grafikzeichen definiert und über den Bildschirm bewegt werden. Ändern Sie Zeile 140 in:

```
140 PRINT U$;:GOTO 140
```

Achten Sie bitte auf das Semikolon vor dem Doppelpunkt. In einer ewigen Schleife wird nun das Rechteck diagonal über den Bildschirm gedruckt. Versuchen Sie es ruhig selbst einmal, verändern Sie die Variablen Q$, R$, S$, oder T$. Mal sehen, was passiert!

Der Vergleich zweier Strings kommt sehr oft vor. Listing 5 zeigt das Prinzip. In Zeile 100 weisen wir der Variablen A$ den String "FLOPPY" zu. Zeile 110 erwartet die Eingabe eines neuen Strings, dem die Variable X$ zugewiesen wird. Der Vergleich folgt in Zeile 120. Erst wenn beide Strings Zeichen für Zeichen identisch sind, wird das Wort »TREFFER« auf dem Bildschirm ausgegeben. In Zeile 130 erfolgt ebenfalls ein Vergleich, nur anders herum. Solange der eingegebene String nicht aus dem einzelnen Zeichen @ besteht, ist die Prüfung auf Ungleichheit erfüllt, und das Programm kehrt zu einer neuen Eingabe (Zeile 110) zurück.



Programmtechnisch ist es sehr empfehlenswert, in einer vergleichenden INPUT-Schleife immer eine Aussprungmöglichkeit wie in Zeile 130 zu schaffen. Es ist schließlich denkbar, daß ohne Vorkenntnisse das Wort FLOPPY nicht erraten wird. Und dann?

Addierte Strings können natürlich ebenfalls auf Gleichheit oder Ungleichheit geprüft werden. Fügen Sie bitte dem Listing 5 folgende Zeile hinzu:

```
125 IF X$=A$+"1541" THEN PRINT X$
```

Diese Prüfung ist erst erfüllt, wenn die Eingabe FLOPPY 1541 lautet, mit Leerzeichen dazwischen.

Beim Stringvergleich mit dem Größer-als- (>) oder Kleiner-als-Zeichen (<) muß der Programmierer wissen, was einen größeren oder kleineren String ausmacht. In Listing 6 werden Strings miteinander verglichen. Anschließend gibt das Programm aus, welcher String größer ist. Experimentieren Sie ein bißchen. Beachten Sie dazu Tabelle 1.

Sehen Sie den Zusammenhang? Der Computer vergleicht die ASCII-Codewerte der einzelnen Buchstaben von links aus. ASCII-Codes sind Zahlen, mit denen der Computer intern ein Zeichen kennzeichnet. Dazu später mehr. Das A hat den ASCII-Wert 65, B den Wert 66, C den Wert 67. Sowohl bei ACB als auch bei ABC ist das erste Zeichen gleich, aber von den zweiten Zeichen ist C größer als B. Im Beispiel TISCHE-TISCH macht das zusätzliche E im String #1 den Unterschied. Tabelle 2 ist eine vollständige Auflistung der ASCII-Codes, die Sie zum Vergleich heranziehen können.

Subtrahieren von Strings mit dem Minuszeichen geht leider nicht. Basic bietet uns jedoch drei Befehle, die uns gestatten, einzelne Zeichen eines Strings abzuschneiden beziehungsweise herauszupicken. LEFT$ schneidet vom linken Rand des Strings Zeichen heraus, RIGHT$ macht dasselbe auf der rechten Seite. MID$ pickt Teile aus der Mitte

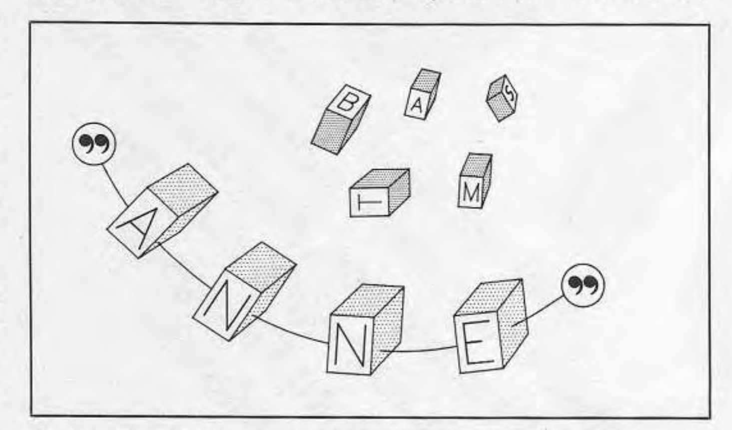


Bild 1. Wie ein Namenskettchen werden Zeichen und Buchstaben zwischen Gänsefüßchen zu Strings zusammengefaßt

String # 1	String # 2	Wer ist größer?
A	B	B
ACB	ABC	ACB
TISCHE	TISCH	TISCHE
21	26	26
DDD	DDD	ohne Wirkung
WORT 2	WORT 1	WORT 2

Tabelle 1. Strings besitzen Codewerte, anhand derer sie sich in ihrer Größe beziehungsweise Wertigkeit unterscheiden lassen. Hier sehen Sie einige Beispiele.

heraus. Zusätzlich - aber nur bei C 16/116, Plus/4 und C 128 - fügt er auch Teile in die Mitte eines Strings ein.

Natürlich können wir bestimmen, wie viele Zeichen abgetrennt werden sollen. LEFT$ und RIGHT$ benötigen nur eine einzige, MID$ zwei Zahlenangaben. Im Beispiel (Listing 7) wird das schnell klar. In Zeile 20 beginnen wir an der linken Seite des Wortes. Hinter LEFT$ steht in der Klammer zuerst der String, um den es geht. In diesem Fall ist es A$. Die Zahlenangabe nach dem Komma bestimmt die Anzahl der abzuschneidenden Zeichen. Im Beispiel sind es fünf.

Welcher String ist größer?

Diese fünf ergeben den neuen String »MOTOR«, der in Zeile 30 ausgedruckt wird. Schreibweise und Funktion von RIGHT$ ist identisch, nur daß dieser Befehl von der rechten Seite des Wortes A$ aus wirkt. In Zeile 40 erhält dieser Teilstring den Variablennamen C$. Zeile 70 druckt die rechten sieben Zeichen von A$ aus. Ergebnis ist das Wort »SCHLUSS«. Interessant wird es in Zeile 60 beim Befehl MID$. Der Angabe des betroffenen Strings folgt die Nummer des Zeichens, ab dem, von links gezählt, herausge-

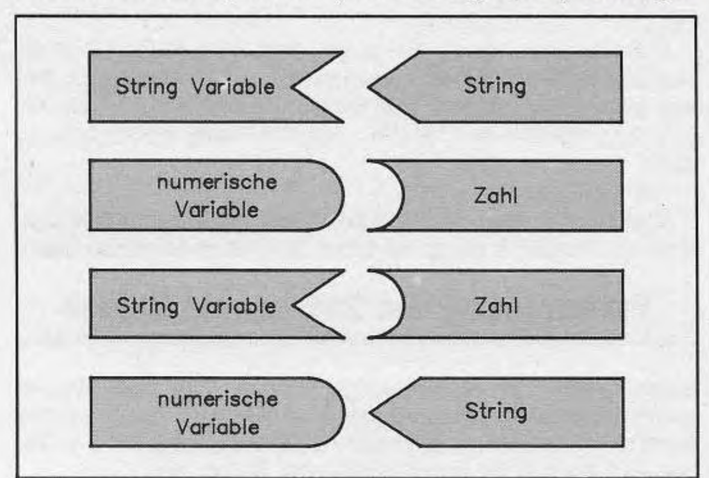


Bild 2. Den Variablen müssen die richtigen Werte zugewiesen werden. Sonst kommt es zu einer Fehlermeldung.

schnipselt werden soll. Die zweite Zahl gibt an, wieviel Zeichen es sein sollen. In unserem Beispiel ist das sechste Zeichen von links das »H«. Fünf Zeichen weiter, inklusive H, ergeben das Wort »HAUBE«. In Zeile 70 wird es ausgedruckt.

Bild 3 zeigt die Funktionen der drei Befehle auf einen Blick.

Die Frage stellt sich, ob wir mit diesem Befehl zwei getrennte Teile aus einem Wort herausholen, und sie so zusammensetzen können, daß ein neues Wort entsteht. Wir können! Als Beispiel benutzen wir das Wort »DRACHEN«, aus dem das Wort »RAHE« entstehen soll. Wir picken lediglich die Teile »RA« und »HE« heraus und addieren sie zu einem neuen String. Listing 8 zeigt, wie es geht.

```
100 A$="HOLZ"                <058>
110 B$="FEUER"               <115>
120 PRINT A$ B$              <123>
```

Listing 1. Die Addition ist die einfachste Manipulation von Strings. Subtrahieren ist unmöglich.

```
100 X$="7"                   <051>
110 Y$="5"                   <193>
120 Z$=X$+Y$                 <055>
130 PRINT Z$                 <252>
```

Listing 2. Die Addition von Strings. Die Zahlen sind nicht selbstverständlich.

```
100 A$="HOLZ"                <058>
110 B$="FEUER"               <115>
120 L$="{RVSON}"             <206>
130 M$="{RVOFF}"             <252>
140 PRINT L$+B$+M$+A$        <078>
```

Listing 3. Steuerzeichen können als Strings dargestellt werden. Man erhält einen schönen Effekt.

Es ist nicht zwingend vorgeschrieben, daß die Zahlen in der Klammer hinter den String-Befehlen konstant sind. Geben Sie Listing 9 ein und starten es mit RUN. Dieses Programm druckt das folgende Muster aus:

```
D
DR
DRA
DRAC
DRACH
DRACHE
DRACHEN
```

Das Geheimnis liegt in den Zeilen 110 und 120, in denen die Zahl der abzuschneidenden Zeichen nicht konstant, sondern durch die Zählerschleife mit X von 1 bis 7 hochgezählt wird. Selbstverständlich geht es mit RIGHT$ auch andersrum. Listing 10 liefert uns dieses Bild:

```
DRACHEN
RACHEN
ACHEN
CHEN
HEN
EN
N
```

Die rückwärts zählende Schleife wird durch die Festlegung des Anfangswertes X=7 in Zeile 110 und durch das laufende Vermindern der Zählvariablen in Zeile 130 gebildet. Eine elegantere Lösung ermöglicht der MID$-Befehl. Kehren wir wieder zur hochzählenden Schleife zurück und zwicken von links der Reihe nach die Buchstaben heraus. Listing 11 realisiert unser Vorhaben.

Wenn Sie die Zeile 120 so schreiben:

```
120 B$=MID$(A$,X,1)
```

wird das Wort mit einzelnen Buchstaben untereinander geschrieben. Eine andere Variante sieht so aus:

```
120 B$=MID$(A$,1,X)
```

Dieses Muster ist schon von Listing 9 her bekannt. Ein letztes Arrangement erhalten wir durch die Programmzeile

```
120 B$=MID$(A$,X,X)
```

Diese Variation der beiden Parameter ergibt nachstehendes interessantes Muster:

```
D
RA
ACH
CHEN
HEN
EN
N
```



Legen wir unser Augenmerk auf Zeile 140 des Listings 11. Dort befindet sich eine Zählschleife, die sich so lange wiederholt, bis X den Wert 7 angenommen hat. Dieser Wert entspricht der Anzahl an Buchstaben in dem Wort DRACHEN. Wollen wir mit einem anderen Wort experimentieren, so muß der Variablen X ein anderer Wert zugeordnet werden. Beim Wort »MENSURALNOTATION« (eine im 13. Jahrhundert ausgebildete Notenschrift, die die Tondauer

LEN$ – bequemes Abzählen

angibt) bereitet das Zählen eine Menge Arbeit (es hat 16 Buchstaben). Es ist somit sinnvoll, einen Befehl zu gebrauchen, der die Zählarbeit abnimmt. Basic wartet mit dem schönen Befehl LEN$ auf. Er ist eine Abkürzung des englischen Wortes »length«, das »Länge« bedeutet. Wenn Sie direkt eingeben

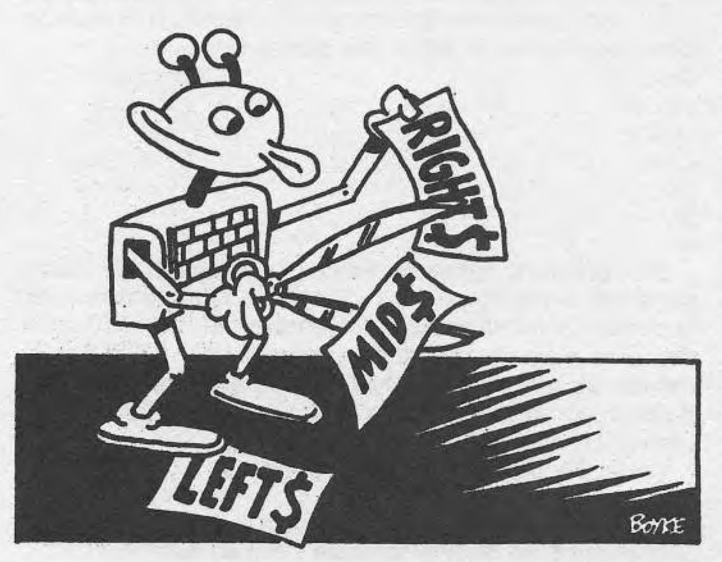
```
PRINT LEN("MENSURALNOTATION")
```

erhalten sie den Wert 16. Erhalten Sie einen anderen Wert, haben Sie sich verschrieben. Um diesen Befehl in unser letztes Programm einzubauen, ändern Sie Zeile 140 in:

```
140 IF X<>LEN(A$) THEN 110
```

Sie prüft X, bis es den Wert von LEN(A$) erreicht hat.

Beim Größenvergleich von Strings haben wir gesehen, daß der Computer mit ASCII-Codes arbeitet. Alle Computer verwenden intern irgendwelche Code-Zahlen, um Zeichen, Buchstaben und Zahlen im Rechenwerk, Speicher oder Peripherie darzustellen. Theoretisch kann jeder Hersteller diese Codes definieren, wie er will. Beim Datenaustausch mit anderen Computern oder mit einem Drucker müssen die Daten aber einem international standardisierten Code entsprechen. Dieser Standard heißt »American Standard for Information Interchange«, abgekürzt ASCII. Jedes Zeichen, jede Zahl und jede Funktion hat darin einen eigenen Code-Wert.



Strings und der ASCII-Code

Es ist ein leichtes, ein Programm zu schreiben, das uns die Abfrage aller ASCII-Codes gestattet. Dabei hilft uns ein Befehl, der einen String – oder genauer gesagt – das erste Zeichen eines Strings in den entsprechenden ASCII-Wert umwandelt – ASC(A$). Listing 12 zeigt seine Wirkungsweise.

Der Pfiff dieses Vierzeilers liegt im neuen Befehl ASC$ in Zeile 20. Er wandelt das eingegebene Zeichen in seinen ASCII-Wert um, der in Zeile 30 zusammen mit dem Zeichen ausgedruckt wird. Vorsicht! Geben Sie den ASCII-Code eines Steuerzeichens ein, wird dessen Funktion durch Zeile 30 ausgeführt.

```
100 Q$="QYP"                  <027>
110 R$="{3LEFT}"              <093>
120 S$="{DOWN}"               <170>
130 T$="LPQ"                  <107>
140 U$=Q$+R$+S$+T$            <081>
150 PRINT U$                  <232>
```

Listing 4. Grafikzeichen können zu Strings addiert und über den Bildschirm bewegt werden

```
100 A$="FLOPPY"                      <204>
110 INPUT "EINGABE";X$               <250>
120 IF X$=A$ THEN PRINT"TREFFER"     <088>
130 IF X$<>"@"THEN 110               <107>
140 END                              <142>
```

Listing 5. Eine Routine, die sehr oft vorkommt: der Vergleich zweier Strings

```
100 INPUT "STRING #1";A$             <094>
110 IF A$="@" THEN END               <089>
120 INPUT "STRING #2";B$             <186>
130 IF A$>B$ THEN PRINT "#1"         <166>
140 IF A$<B$ THEN PRINT "#2"         <000>
150 GOTO 100                         <078>
```

Listing 6. Strings unterscheiden sich in ihrer Größe. Experimentieren Sie ein wenig.

Die Umkehrung des ASC-Befehls ist der CHR$(X)-Befehl (sprich: Character-String). Der Name ist die Abkürzung von Character, was soviel heißt wie »Zeichen«. Er wandelt die ASCII-Zahl X in ihr entsprechendes Zeichen um. In Verbindung mit dem PRINT-Befehl kann er Zeichen und Buchstaben auf den Bildschirm bringen:

```
PRINT CHR$(65) <RETURN>
```

bewirkt dasselbe wie

```
PRINT"A" <RETURN>
```

Gerade eben wurde davor gewarnt, den ASCII-Code eines Steuerzeichens einzugeben. Natürlich kann man dieses gezielt ausnützen. Der Bildschirm kann zum Beispiel mit dem ASCII-Wert für die <CLR>-Taste gelöscht werden:

```
PRINT CHR$(147)
```

Vielleicht fragen Sie sich an dieser Stelle, was das soll. Nun, im Vergleich zur guten alten Gänsefuß-Methode bietet

Verwandlung von Strings und Zahlen

dieses Verfahren schon einige Vorteile. Die ASCII-Werte sind zum einen viel besser druck- und lesbar, zum anderen kann man mit Zahlen, auch mit ASCII-Werten, rechnen. Geben Sie doch bitte einmal folgende Zeilen ein:

```
10 X=64
20 X=X+1
30 PRINT CHR$(X)
40 IF X<90 THEN 20
```

In Zeile 10 geben wir der numerischen Variablen X den Wert 64, das ist um 1 weniger, als der ASCII-Wert des Buchstabens A.

In Zeile 20 wird eine Zählschleife begonnen mit der Erhöhung von X um 1.

Zeile 30 druckt mit dem CHR$-Befehl das dieser Zahl entsprechende Zeichen aus. Im ersten Durchlauf ist X=65, das ist der ASCII-Wert für das A.

Zeile 40 schließt die Zählschleife durch den Rücksprung auf Zeile 20. Dadurch werden alle Zeichen vom Wert 65 bis 90 – das ist das Alphabet – ausgedruckt.

Mit Buchstaben in Gänsefüßchen wäre das Programm recht lang geworden. Überzeugt?



Wir haben bereits gelernt, daß Zahlen als Strings, also als reine Ziffernfolge, verarbeitet werden können, wenn sie in Gänsefüßchen stehen. Der Vorteil dieser Methode: Zahlen werden ohne Freiraum für ein eventuelles Vorzeichen ausgegeben. Das kleine Programm

```
10 A$="123"
20 PRINT A$
```

druckt die Zahl ganz an den linken Rand. Es gibt zwei Befehle, die uns erlauben, Strings in Zahlen und Zahlen in Strings umzuwandeln. VAL(A$) wandelt A$ in einen Zahlenwert um, falls dort eine Zahl vorkommt. Die Zeile

```
30 PRINT VAL(A$)
```

druckt ebenfalls die Zahl 123 aus, aber eben als Zahl, das heißt mit einer Leerstelle vor ihr. Ist in dem String keine Zahl enthalten, wird der Wert Null ausgegeben:

```
40 B$="ABC"
50 PRINT VAL(B$)
```

Schreiben wir statt "ABC" den String "A2C", erhalten wir immer noch Null als Ergebnis, weil der String mit einem

```
10 A$="MOTORHAUBENVERSCHLUSS"     <213>
20 B$=LEFT$ (A$,5)                <197>
30 PRINT B$                       <216>
40 C$=RIGHT$ (A$,7)               <005>
50 PRINT C$                       <244>
60 D$=MID$(A$,6,5)                <201>
70 PRINT D$                       <016>
```

Listing 7. Worte können geteilt und die einzelnen Teilstücke zu neuen Strings definiert werden

```
100 A$="DRACHEN"                  <109>
110 X$=MID$(A$,2,2)               <011>
120 Y$=MID$(A$,5,2)               <037>
130 Z$=X$+Y$                      <065>
140 PRINT Z$                      <006>
```


Listing 8. Aus dem Wort »DRACHEN« machen wir das Wort »RAHE« mit Hilfe von Strings

```
100 A$="DRACHEN"                  <109>
110 X=X+1                         <138>
120 B$=LEFT$(A$,X)                <182>
130 PRINT B$                      <060>
140 IF X<>7 THEN 110              <224>
```

Listing 9. Die Zahlen hinter den String-Befehlen müssen nicht konstant sein. Ein interessantes Muster entsteht.

Buchstaben anfängt. Ist B$ jedoch "12C", ergibt der VAL$-Befehl die Zahl 12. Diese Eigenschaft läßt sich oft sinnvoll in einem Programm einsetzen.

Die Umkehrung von VAL$ ist STR$(X), abgeleitet von String. Er wandelt die Zahl X in einen String um. Wozu das gut ist, zeigen uns die nächsten Zeilen:

```
10 X=123
20 Y=456
30 PRINT X+Y
```

Das Resultat ist die Summe beider Zahlen, also 579, mit einer Leerstelle vor der neuen Ziffer ausgedruckt.

Die Zeile:

```
40 PRINT X;Y
```

setzt dagegen beide Zahlen nebeneinander, getrennt durch zwei Leerstellen. Eine für das Vorzeichen, die zweite für die Trennung zweier unabhängiger Zahlen:

```
123 456
```

Diese zweite Trennung heben wir mit dem STR$-Befehl auf:

```
50 PRINT STR$(X),STR$(Y)
```

Wir erhalten:

```
123 456
```

X und Y sind aber immer noch Zahlen, obwohl sie über STR$ als Strings behandelt werden. Erst die Zeile:

```
60 PRINT "123";"456"
```

macht echte Strings aus ihnen und vermeidet alle Zwischenräume. Leider kann man in dieser Version mit den Zahlen nicht mehr rechnen. Mit STR$ geht es aber, wie die nächsten Zeilen zeigen:

```
65 FOR Z=0 TO 2
70 PRINT STR$(X+Z);STR$(Y)
75 NEXT Z
```

```
100 A$="DRACHEN"                  <109>
110 X=7                           <215>
120 B$=RIGHT$(A$,X)               <214>
130 PRINT B$                      <060>
140 X=X-1                         <200>
150 IF X<>0 THEN 120              <043>
```

Listing 10 ist eine Umkehrung von Listing 9. Es besitzt eine rückwärtszählende Schleife.

```
100 A$="DRACHEN"                  <109>
110 X=X+1                         <138>
120 B$=MID$(A$,X,8-X)             <176>
130 PRINT B$                      <060>
140 IF X<>LEN(A$) THEN 110        <160>
```

Listing 11. Eine elegante Lösung des Problems von Listing 10 ermöglicht der MID$-Befehl

Wir erhalten die Zahlenreihen

```
123 456
124 456
125 456
```

STR$ und VAL$ – wirklich nützlich?

Ein Zweifel an der Nützlichkeit dieser beiden Befehle scheint nicht unbegründet. Tatsächlich sind sie nicht so gängig, wie andere String-Befehle. Ein paar kleine Beispiele sollen jedoch den Zweifel widerlegen. Mit dem LEN-Befehl haben wir schon die Länge eines Strings festgestellt. Mit Zahlen geht das normalerweise nicht. Versuchen Sie es ruhig einmal mit den Zeilen:

```
10 A=1234
20 PRINT A
```

Wollen wir die Anzahl der Ziffern von A feststellen, und machen es so:

```
30 PRINT LEN(A)
```

werden wir Schiffbruch erleiden. Der Computer weist uns mit einem TYPE MISMATCH ERROR zurecht. Also muß zuerst STR$ die Zahl in einen String umwandeln:

```
30 PRINT LEN(STR$(A))
```

Als Resultat erhalten wir die Zahl 5. Warum 5? Nun, die Vorzeichenstelle wird mitgezählt. Wenn man das weiß, kann man sie ja vom Ergebnis abziehen. Für eine Darstellung auf dem Bildschirm ist das mitunter wichtig.

Beim Schreiben von Tabellen und Zahlenkolonnen nehmen die Vorzeichen-Leerstellen oft unnötig viel Platz weg. Ein Ärgernis für viele Programmierer. Mit dem STR$-Befehl kann dieser Platz abgezogen werden. Versuchen Sie mal, die Zahl 200 zwischen die beiden Striche zu schreiben, die mit der <SHIFT>-<B>-Taste erzeugt werden.

```
100 PRINT "I"200"I"
```

mit RUN erhalten wir:

```
I 200 I
```

Die Schreibweise in Zeile 100 erzeugt die Leerstellen. Die Zeile 110 löst das Problem:

```
110 PRINT "I"MID$(STR$(200),2)"I"
```



Sehen Sie den Trick? Mit STR$(200) bilden wir aus der Zahl 200 einen String " 200", also mit Leerstellen. Diesen String schreiben wir mit dem MID$-Befehl erst ab der zweiten Stelle zwischen die beiden Striche, ohne Leerstelle. In späteren Teilen dieses Kurses werden wir mit STR$ noch viel arbeiten.

Wecker mit VAL$

Ein kleines »Weckerprogramm« stellt den VAL$-Befehl vor. Wir werden zehn Sekunden lang den Bildschirm mit Sternen füllen. Die eingebaute Uhr des Computers hilft uns dabei. Sie kann über die Uhr-Variable TI$ auf Null gesetzt werden. Geben Sie Listing 13 ein.

Zeile 100 definiert die Laufzeit Z. In Zeile 110 setzen wir, wie im Commodore-Handbuch beschrieben, die Uhr auf die Zeit 00 Stunden 00 Minuten 00 Sekunden, mit der die Uhr sofort weiterläuft. Es werden über Zeile 120 so lange Sterne gedruckt, bis in Zeile 130 der Wert des Strings TI$ mit Z=10 übereinstimmt. Anschließend wird abgebrochen, und die Nachricht der Zeile 140 ausgedruckt. Sie könnten auch statt dieser Nachricht eine Alarmglocke läuten lassen. Das liegt bei Ihnen.

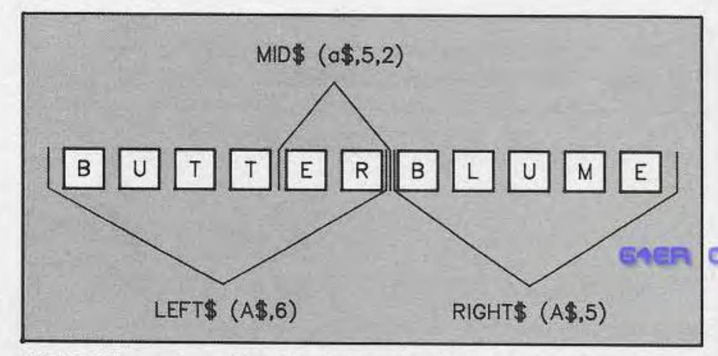


Bild 3. Hier sehen Sie die drei Basic-Befehle, mit denen man Strings zerteilen kann, auf einen Blick

Damit sind wir bereits am Ende unseres ersten Teils des String-Kurses angelangt. Außer INSTR, PRINT USING und PUDEF wurden alle wichtigen Befehle abgehandelt. Da sie aber einer ausführlichen Erklärung bedürfen und nur beim C 128, C 16/116 und PLUS/4 vorkommen. Lassen wir diese Befehle in unserem Kurs unberücksichtigt. Bis zum weiteren Teil wollen Sie vielleicht erst einmal eine kleine Pause einlegen. Spielen Sie ruhig mit Strings ein wenig rum, verwirklichen Sie eigene Ideen.

2. Texteingabe und Textverarbeitung mit Strings

Jeder Einsteiger lernt sehr schnell, Überschriften und Text mit dem Print-Befehl in seine Programme einzubauen. Daß Basic aber sehr viel mehr Möglichkeiten bietet, Texte und Zeichenketten (Strings) zu verarbeiten, wird klar, wenn man andere Programme sieht, in denen Strings verglichen, verändert, bewegt oder auf andere Weise manipuliert werden. Auf Anhieb ist trotz Kenntnis der verwendeten Befehle nicht immer klar, was da passiert. Im ersten Teil haben wir bereits alle wesentlichen String-Befehle und String-Funktionen behandelt und mit kleinen Beispielen vorgeführt. Diesmal nehmen wir uns das Thema Texteingabe vor.

In fast allen Programmen wird der Benutzer um Angaben zum Programmverlauf gebeten, die über die Tastatur eingegeben werden müssen. Oft sind es Zahlen, die gefragt sind, meistens aber sind es Texte oder einzelne Wörter.

2	3	4	5	6	7	8
SPACE 032	@ 064	096	128	SHFT SPACE 160	192	SHFT SPACE 224
! 033	A 065, a 065	097, A 097	BLK 129	161	193, A 193	225
" 034	B 066, b 066	098, B 098	130	162	194, B 194	226
# 035	C 067, c 067	099, C 099	LOAD RUN 131	163	195, C 195	227
$ 036	D 068, d 068	100, D 100	132	164	196, D 196	228
% 037	E 069, e 069	101, E 101	F1 133	165	197, E 197	229
& 038	F 070, f 070	102, F 102	F3 134	166	198, F 198	230
' 039	G 071, g 071	103, G 103	F5 135	167	199, G 199	231
(040	H 072, h 072	104, H 104	F7 136	168	200, H 200	232
) 041	I 073, i 073	105, I 105	F2 137	169, 169	201, I 201	233, 233
* 042	J 074, j 074	106, J 106	F4 138	170	202, J 202	234
+ 043	K 075, k 075	107, K 107	F6 139	171	203, K 203	235
, 044	L 076, l 076	108, L 108	F8 140	172	204, L 204	236
- 045	M 077, m 077	109, M 109	SHFT RET 141	173	205, M 205	237
. 046	N 078, n 078	110, N 110	GRAF 142	174	206, N 206	238
/ 047	O 079, o 079	111, O 111	143	175	207, O 207	239
0 048	P 080, p 080	112, P 112	BLK 144	176	208, P 208	240
1 049	Q 081, q 081	113, Q 113	CRSR UP 145	177	209, Q 209	241
2 050	R 082, r 082	114, R 114	RVS OFF 146	178	210, R 210	242
3 051	S 083, s 083	115, S 115	CLR SCRN 147	179	211, S 211	243
4 052	T 084, t 084	116, T 116	INS 148	180	212, T 212	244
5 053	U 085, u 085	117, U 117	BRN 149	181	213, U 213	245
6 054	V 086, v 086	118, V 118	LT RED 150	182	214, V 214	246
7 055	W 087, w 087	119, W 119	DARK GREY 151	183	215, W 215	247
8 056	X 088, x 088	120, X 120	MED GREY 152	184	216, X 216	248
9 057	Y 089, y 089	121, Y 121	LT GRN 153	185	217, Y 217	249
: 058	Z 090, z 090	122, Z 122	LT BLU 154	186, 186	218, Z 218	250, 250
; 059	[091	123	LT GREY 155	187	219	251
< 060	£ 092	124	PURP 156	188	220	252
= 061	] 093	125	CRSR LEFT 157	189	221	253
> 062	↑ 094	126, 126	YEL 158	190	222, 222	254
? 063	← 095	127, 127	CYAN 159	191	223, 223	255, [illegible]

Tabelle 2. ASCII-Code-Tabelle aller Zeichen

Dem Programmierer steht dafür der INPUT-Befehl zur Verfügung. Vorteilhaft ist, daß dieser Befehl einen Hinweis enthalten kann, was eingegeben werden soll.

```
10 INPUT "TEXT";A$
20 PRINT A$
```

Diese Eingabemethode funktioniert gut, hat aber ein paar Eigenheiten:

- Komma und Doppelpunkt dürfen im einzugebenden Text nicht vorkommen
- ein Zeilensprung mit <RETURN> ist nicht möglich, da diese Taste den INPUT-Vorgang beendet
- der Text, einschließlich Leerstellen, darf nicht länger als 88 Zeichen sein
- der Eingabevorgang kann nicht mit der STOP-Taste abgebrochen werden.

Einige Beispiele:

Starten Sie die Zeilen 10 und 20 mit RUN. Wenn Sie hinter dem blinkenden Fragezeichen eingeben:

```
WERT:ZAHL
```

oder

```
WERT,ZAHL
```

erscheint eine Fehlermeldung

```
EXTRA IGNORED WERT
```

Der Text ab dem Komma beziehungsweise ab dem Doppelpunkt wird unterschlagen.

Die Eingabe:

"WERT" IST

führt zur Fehlermeldung REDO FROM START.

Bei einer Eingabe:

"DER WERT

wird das Gänsefüßchen nicht geschrieben.

Dagegen wird die folgende Eingabe akzeptiert:

DER "WERT" IST .

Sie sehen, man muß aufpassen.

Will man diese Einschränkungen umgehen, verwendet man den GET-Befehl. Allerdings kann mit GET, wie Sie vielleicht schon wissen, immer nur ein einziges Zeichen eingegeben werden. Der GET-Befehl benötigt jedoch nicht die Betätigung der RETURN-Taste, eignet sich also für unsere Zwecke wesentlich besser.

```
10 GET A$:IF A$="" THEN 10                <115>
20 A=ASC(A$)                              <170>
30 PRINT A$,A                             <165>
40 GOTO 10                                <218>
```

Listing 12 wandelt einen eingegebenen Buchstaben in seinen ASCII-Code um

```
100 Z=10                                  <123>
110 TI$="000000"                          <089>
120 PRINT "*";                            <163>
130 IF VAL(TI$)<>Z THEN 120               <186>
140 PRINT:PRINT "10 SEKUNDEN"             <190>
```

Listing 13. Ein kleines »Weckerprogramm« demonstriert die Funktion des VAL$-Befehls

Für die Eingabe eines längeren Strings muß eine Schleife gebildet werden, die den GET-Befehl für jedes einzelne Zeichen des Strings aufruft. Die folgenden drei Zeilen zeigen dies:

```
110 GET A$:IF A$=""
    THEN 110
120 PRINT A$;
150 GOTO 110
```

In Zeile 110 erwartet der Computer eine Eingabe über Tastatur. GET A$ bewirkt ein ständiges Abfragen. Solange keine Eingabe erfolgt, bleibt das Programm in Zeile 110. Zeile 120 gibt das eingegebene Zeichen auf Bildschirm aus. Zeile 150 springt wieder zurück zu 110; eine unendliche Schleife. Besitzer eines C 16/116 und eines C 128 schreiben die Zeile 110 so:

```
110 GETKEY A$
```

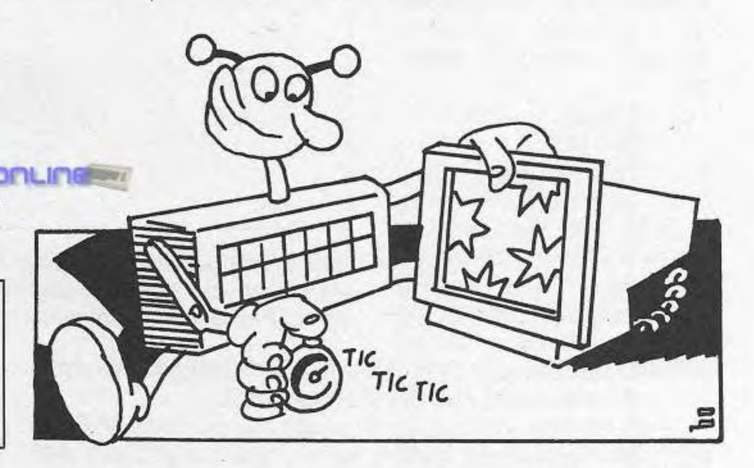
Jetzt können wir eingeben, was wir wollen: Komma, Zeilensprünge mit <RETURN> und so weiter. Nachteilig ist, daß der ganze schöne geschriebene Text nicht gespeichert werden kann. Unter der Variablen A$ merkt sich der Computer immer nur das zuletzt eingegebene Zeichen. Wir müssen also die Zeichen in einer neuen String-Variablen zusammenbinden.

Und wieder String-Addition

Die String-Addition kommt uns zu Hilfe. Die »Sammelvariable« für den Text soll T$ heißen. Sie setzt sich zusammen aus dem jeweiligen Text T$ plus dem mit GET neu eingegebenen Buchstaben oder Zeichen A$ (Zeile 130). Geben Sie folgende Zeilen ein und starten sie mit RUN.

```
110 GET A$:IF A$=""
    THEN 110
120 PRINT A$;
130 T$=T$+A$
150 GOTO 110
```

Im ersten Durchlauf steht in T$ noch nichts, aber der erste eingegebene Wert für A$ wird durch den Rücksprung der Zeile 150 in T$ eingesetzt. So geht das munter weiter.



Zeile 120 druckt den zuletzt getippten Buchstaben A$ aus. Das Semikolon sorgt dafür, daß alle Buchstaben in einer Reihe stehen.

Die einzige Beschränkung besteht darin, daß einer String-Variablen nur maximal 255 Zeichen zugeordnet werden dürfen, aber das ist schon eine ganze Menge.

1. Die Texteingabe mit INPUT ist auf 88 Zeichen beschränkt. Der Text darf kein Komma oder Doppelpunkt enthalten; bei Gänsefüßen ist Vorsicht geboten. Die RETURN-Taste schließt die Eingabe ab.
2. Die Texteingabe mit einer GET-Schleife läßt alle Zeichen zu. Da mit GET aber immer nur ein einziges Zeichen eingebbar ist, muß ein längerer Text durch String-Addition zusammengesetzt werden.
3. Der Inhalt einer String-Variablen kann maximal 255 Zeichen betragen.

Wie schon erwähnt, sind jetzt alle Tasten erlaubt, auch die Cursor-Tasten, ja sogar die RETURN-Taste, die einen Zeilensprung im Text auslöst. Da <RETURN> uns aber nicht, wie bei INPUT, als Zeichen der Beendigung der Eingabe zur Verfügung steht, kommen wir normal nicht aus der Schleife heraus, nur die STOP-Taste schafft es, was leider aber zum Abbruch des Programms führt.

Wir brauchen eine andere Taste, mit der die Eingabe des Strings abgeschlossen werden soll. Sie darf natürlich im

normalen Text nicht vorkommen. Zur Lösung dieses Problems hilft uns dabei der ASC(A$)-Befehl. Er bestimmt den ASCII-Code eines eingegebenen Zeichens A$. Sie erinnern sich bestimmt an meine Erklärung des ASCII-Codes im ersten Teil dieses Kurses; wenn nicht, dann bitte noch mal nachlesen.

Eine nicht im Text vorkommende Tastenkombination ist zum Beispiel <SHIFT> und <RETURN> gleichzeitig gedrückt. Sie hat einen anderen ASCII-Code als die RETURN-Taste allein, nämlich 141.

Mit dieser Tastenkombination wollen wir die Beendigung der Eingabe signalisieren. Wir prüfen sie in Zeile 140 nach, vor dem Rücksprung auf Zeile 110:

```
140 IF ASC(A$)=141
    THEN 160
150 GOTO 110
160 PRINT CHR$(147) T$
```

Ist die Eingabe beendet, wird in unserem Beispiel durch Zeile 160 der gesamte aufaddierte Text T$ ausgedruckt. Zur besseren Übersicht erst, nachdem mit CHR$(147) der Bildschirm gelöscht wurde.

Das GOTO in Zeile 150 können wir vermeiden, wenn wir diese Zeilen wesentlich eleganter schreiben:

```
140 IF ASC(A$) <> 141
    THEN 110
150 PRINT CHR$(147) T$
160  entfällt.
```

Zeile 140 verzweigt so lange zu Zeile 110, bis Sie <SHIFT> und <RETURN> gleichzeitig eingeben. Anschließend führt sie zu Zeile 150. Die alte Zeile 150 wird überflüssig. Unsere alte Zeile 160 bekommt nun die Nummer 150.

Wem die Beendigung der Eingabe mit der geSHIFTeten RETURN-Taste nicht gefällt, kann natürlich in Zeile 140 auch andere Tasten abfragen. Es empfiehlt sich allerdings nicht, eine Taste oder Tastenkombination zu nehmen, die eine Steuerfunktion – zum Beispiel CURSOR-DOWN – ausführt. Es gibt eine ganze Reihe von Tastenkombinationen, die einen eigenen ASCII-Codewert haben. Sie stehen allerdings nicht im Handbuch. Es sind Kombinationen mit der CTRL-Taste.

Mit den drei folgenden Programmzeilen können Sie selbst diese ASCII-Codewerte abfragen:

```
10 GET A$:IF A$="" THEN 10
20 PRINT ASC(A$)
30 GOTO 10
```

Zeile 20 druckt den ASCII-Code der gedrückten Taste oder Tastenkombination aus. Welche ASCII-Werte Sie mit der CTRL-Taste erhalten und für eine Abfrage verwenden können, steht in Tabelle 3.

Wir haben 34 Tastenkombinationen mit eigenen ASCII-Codewerten zur Verfügung, die wir zur Abfrage von gedrückten Tasten verwenden können. Einige sind mit Vorsicht zu genießen. CTRL-E zum Beispiel hat denselben ASCII-Codewert wie die Farbe "WEISS", und sie erzeugt auch diese Farbe unter dem Cursor. CTRL-R produziert reverse Zeichen, CTRL-N schaltet auf den zweiten Zeichensatz mit Kleinbuchstaben um. Ich rate Ihnen daher, nur CTRL-Kombinationen zu wählen, die in Tabelle 3 als leer angegeben sind. Empfehlenswert ist, die ASCII-Tabelle in Ihrem Handbuch mit den zusätzlichen Werten zu ergänzen.

In unserem kleinen Programm wollen wir jetzt in Zeile 140 das Ende der Eingabe mit der Tastenkombination CTRL-A abfragen:

```
140 IF ASC(A$) <> 1 THEN 110
```

Die VC 20-Besitzer haben hier ein Problem. Die Kombination CTRL-A hat laut obiger Tabelle keinen ASCII-Code. Ich empfehle, die Funktionstasten zu nehmen. Die F1-Taste zum Beispiel hat den ASCII-Code 133. Die Beendigungszeile lautet dann für den VC 20:

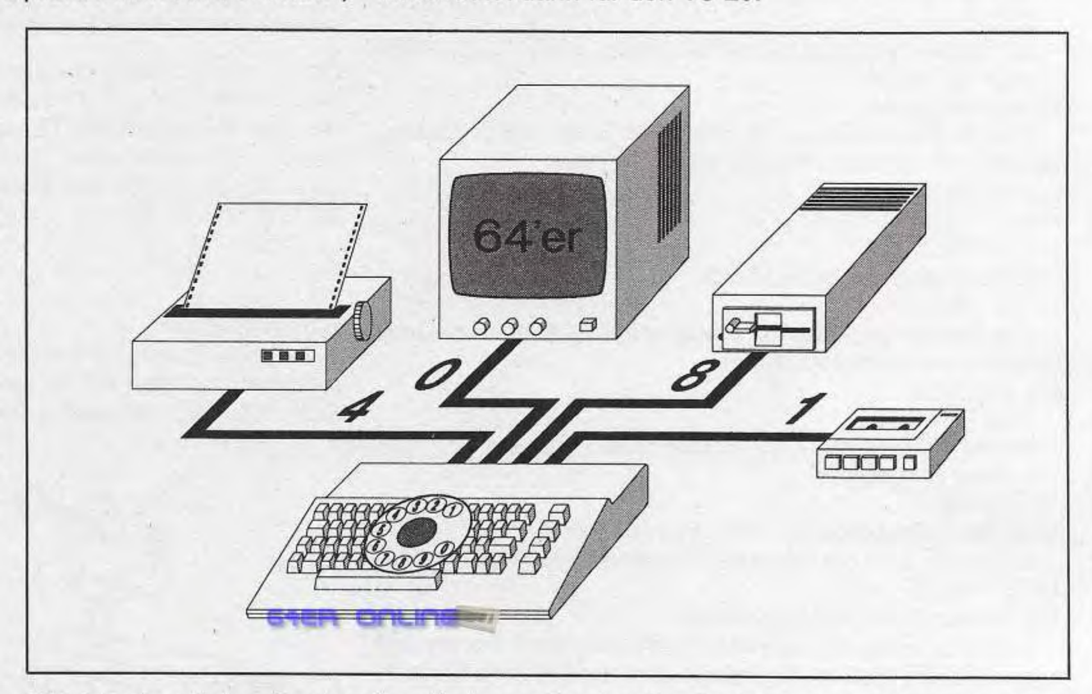


Bild 4. Erst nachdem die Eingabe mit einer vorher bestimmten Taste beendet ist, werden die Strings auf dem Bildschirm im oberen Speicherbereich abgelegt. Von dort können sie an ein Peripherie-Gerät weitergegeben werden. In unserem Fall an einen Drucker.

```
140 IF ASC(A$) <> 133 THEN 110
```

Die Funktionstasten bieten sich beim VC 20 und C 64 für Abfragen an. Leider gilt das nicht für C 16 und C 128. Die Funktionstasten sind mit Befehlen belegt. Man müßte diese Befehle erst löschen. Das ist hier zu kompliziert.

Es gibt vielleicht aber auch andere Tastenkombinationen, die Ihnen mehr zusagen. Experimentieren Sie also mit den Kombinationen.

Der künstliche Cursor

Wenn wir schon bei Tricks sind: die Eingabe mit GET läßt den blinkenden Cursor vermissen. Ihn können wir künstlich erzeugen, indem wir die GET-Warteschleife der Zeile 140 erweitern. Die Benutzer von C 16 und C 128 bitte ich, statt GETKEY auch diese Änderung vorzunehmen.

Vor dem Rücksprung auf Zeile 110 drucken wir, im Fall, daß keine Taste gedrückt worden ist, ein spezielles Cursorzeichen – zum Beispiel das Zeichen "*" – gefolgt vom Steuerzeichen CURSOR-LINKS (Zeile 114). Nach einer kurzen Zeitverzögerung in Zeile 115 wird in der nächsten Zeile 116 durch Drucken eines Leerzeichens der künstliche Cursor wieder gelöscht. Eine weitere Zeitverzögerung in Zeile 117 läßt den künstlichen Cursor regelmäßig blinken. Vorsicht, das Semikolon nicht vergessen! Erst in Zeile 118 folgt die Prüfung, ob eine Taste gedrückt worden ist. Probieren Sie dieses Programm einmal aus, um die Wirkungsweise der künstlichen Cursors zu verstehen.

Unser Programm sieht jetzt mit allen Änderungen folgendermaßen aus:

```
110 GET A$
114 PRINT"*{CRSR LINKS}";
115 FOR T=0 TO 100:NEXT T
116 PRINT" {CRSR LINKS}";
117 FOR T=0 TO 100:NEXT T
118 IF A$=""THEN 110
120 PRINT A$;
130 T$=T$+A$
140 IF ASC(A$) <> 1 THEN 110
150 PRINT CHR$(147) T$
```

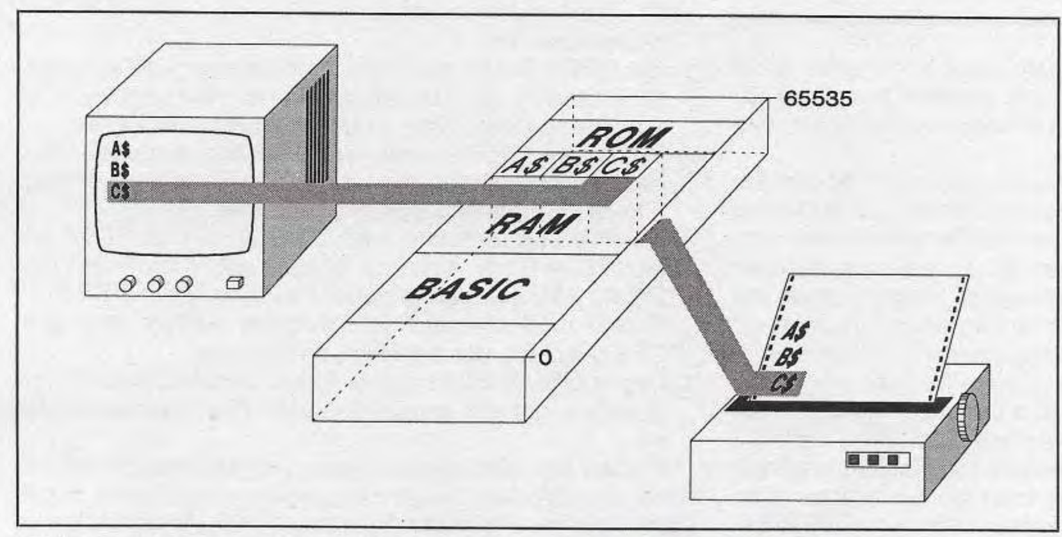


Bild 5. So wie beim Telefon muß mit dem Computer auch eine Verbindung zu anderen »Gesprächspartnern« angewählt werden, bevor es zur Kommunikation kommt.

Tastenkombination	C 128	C 16/116	C 64	VC 20
CTRL-A	1	1	1	
CTRL-B	2	2	2	
	und so weiter			
CTRL-Q	17	17	17	
CTRL-R	18	18	18	18
	und so weiter			
CTRL-Z	26	26	26	
CTRL-:	27	27	27	
CTRL-Pfund	28	28	28	
CTRL-;	29	29	29	
CTRL-,		130		
CTRL-.		132		
CTRL-^		30	30	
CTRL-=	31	6	31	
CTRL-	6		6	6

Tabelle 3. Auch Tastenkombinationen mit <CTRL> haben eigene ASCII-Werte, die abgefragt werden können.

Mit diesem Programm können Sie alle Zeichen der Tastatur eingeben und mit den bisherigen Kenntnissen je nach Bedarf entsprechend verarbeiten.

Wenn Sie die Auswahl reduzieren wollen, zum Beispiel nur auf die Buchstaben A bis Z, müssen Sie eine weitere Prüfzeile 119 einfügen:

```
119 IF ASC(A$) <65 OR ASC(A$) >90 THEN 110
```

Diese Zeile sperrt alle Zeichen, deren ASCII-Codes kleiner als 65 (A) oder größer als 90 (Z) sind. Das erreichen wir mit der OR-Funktion. Durch Erweiterung dieser Formel lassen sich andere Tasten mit einschließen, durch Einklammern des bisherigen Ausdrucks und Anhängen der Zusatzprüfung mit der AND-Funktion:

```
119 IF (ASC(A$) <65 OR ASC(A$) >90)
    AND ASC(A$)<>13 THEN 110
```

Als zusätzlich erlaubte Taste habe ich in der Zeile 119 die RETURN-Taste gewählt - ihr ASCII-Code ist 13.

Wie gesagt, die Klammer nach dem IF und vor dem AND ist wichtig, da das AND beide Grenzwerte - <65 OR >90 - einschließen muß. Probieren Sie es doch einmal mit und ohne Klammer aus, dann sehen Sie den Unterschied.

Drucker als Schreibmaschine

Es wird langsam Zeit, daß wir uns eine echte Anwendung dieser Methode zur langen Texteingabe überlegen. Ich möchte einen Drucker als Schreibmaschine nutzen.

Die Idee kam, als ich meinen ersten Drucker besaß, aber kein Textverarbeitungsprogramm. Ich wollte jedoch schon den Drucker zumindest als Schreibmaschine verwenden.

Ich will dieses Programm, das Sie komplett als Listing 14 vorfinden, mit Ihnen in einzelnen Schritten entwickeln. Tippen Sie also bitte die Teile wie angegeben ein und probieren Sie deren Effekt und Wirkung immer gleich aus.

Nehmen Sie bitte in Kauf, daß die Zeilennummern nicht immer aufeinander folgen werden. Ich verwende die aus Listing 14.

Unser Ziel soll sein, Tastatur und Drucker genau wie eine Schreibmaschine zu betreiben, mit dem Unterschied, daß nicht jeder Buchstabe sofort ausgedruckt wird, sondern zeilenweise. Das erlaubt uns, über den Bildschirm innerhalb einer Zeile vor dem Ausdrucken Korrekturen vorzunehmen (siehe Bild 4).

Zur Benutzung der Tastatur verwenden wir die Eingabe mit GET. Den künstlichen Cursor werden wir wieder verwenden. Da aber die Zeitverzögerung der »Blink-Zeilen« die Geschwindigkeit beeinflußt, mit der neue Buchstaben A$ eingetippt werden können, habe ich die Zeit von dem alten Wert T=100 auf T=30 verkürzt:

```
230 GET A$
234 PRINT"*{CRSR LINKS}";
236 FOR T=0 TO 30:NEXT T
238 PRINT" {CRSR LINKS}";
240 FOR T=0 TO 30:NEXT T
244 IF A$=""THEN 230
262 T$=T$+A$
264 PRINT A$;
282 GOTO 230
```

In Zeile 262 wird aus den einzelnen eingetippten Buchstaben A$ der Text T$ zusammengesetzt. Zur visuellen Kontrolle druckt Zeile 264 jeden einzelnen Buchstaben A$ auf dem Bildschirm aus. Das Semikolon »klebt« sie aneinander und Zeile 282 bildet durch Rücksprung die wiederholbare Schleife.

Wie gesagt, das alles kennen Sie schon. Jetzt kommt der Drucker ins Spiel.

Zur einzelnen Weitergabe von Zeichen und Zahlen an ein Gerät, das an den Drucker angeschlossen ist, müssen wir mit dem OPEN-Befehl eine Verbindung herstellen. Verwechseln Sie das bitte nicht mit dem Einstecken des Verbindungskabels - das ist nur die elektrische Vorausset-

zung. Mit dem OPEN-Befehl wird, wie beim Telefon auch, eine Verbindung »angewählt« (siehe Bild 5).

Wir müssen das immer machen, egal ob es sich um den Drucker, die Datasette oder die Floppystation handelt. Nur der Bildschirm und die Tastatur bilden eine Ausnahme, die sind nämlich sofort nach dem Einschalten des Computers automatisch angewählt.

Ich habe oben mit Absicht gesagt: »Zur einzelnen Weitergabe von Zeichen...« Die Weitergabe von ganzen Programmen geht ohne den OPEN-Befehl mit LOAD und SAVE. Diese Befehle besorgen das »Anwählen« von selbst.

Zeichen für Zeichen

Der Vergleich mit dem Telefon geht sogar noch weiter, denn zum »Anrufen« eines bestimmten Gerätes brauchen wir noch eine »Telefonnummer« - in unserem Fall heißt sie »Gerätenummer«.

Die Nummer der Floppy ist 8, wenn Sie zwei Floppies haben, 8 und 9. Die Datasette hat die Nummer 1, und was uns hier interessiert: Der Drucker hat die Gerätenummer 4.

Der OPEN-Befehl bietet einen weiteren Luxus, nämlich bis zu zehn voneinander unabhängige Verbindungen mit einem Gerät. Diesen einzelnen Verbindungen müssen wir ebenfalls Nummern geben, selbst wenn wir nur eine einzige Verbindung brauchen. Zugelassen sind Zahlen von 1 bis 255. Diese Nummern tragen den schönen Namen »File-Nummer«, weil sie eine Datei - auf englisch »File« - öffnen.

Zum Schluß sei noch eine weitere Möglichkeit erwähnt, die wir hier auch gleich ausnutzen wollen. Neben File-Nummer und Gerätenummer können wir dem OPEN-Befehl noch eine dritte Instruktion mitgeben. Diese heißt »Sekundär-Adresse« und bewirkt Einstellungen oder Umschaltungen im angewählten Gerät. Sie werden gleich sehen, welche Umschaltung wir beim Drucker verwenden.

Als File-Nummer nehmen wir 1, die Gerätenummer des Druckers ist 4, als Sekundär-Adresse nehme ich 7, weil diese Zahl den Drucker in den Klein-/Großbuchstaben-Modus umschaltet, den wir für das Programm brauchen.

```
204 PRINT CHR$(14) CHR$(147)
206 OPEN 1,4,7
```

Zeile 206 ist das Resultat der ganzen Erklärung.

Wir müssen neben dem Drucker auch den Computer in den Klein-/Großbuchstaben-Modus umschalten. Normal machen wir das mit den gleichzeitig gedrückten SHIFT- und COMMODORE-Tasten. Diese Tastenkombination hat den ASCII-Code 14. Zeile 204 zeigt, wie die Umschaltung innerhalb eines Programms mit PRINT und dem CHR$-Befehl gemacht wird. Und weil wir gerade dabei sind, hängen wir in dieser Zeile noch den ASCII-Code für »Bildschirm löschen« (147) an. Nachdem wir jetzt eine Verbindung mit dem Drucker hergestellt haben, wollen wir die einzelnen Zeichen A$ an den Drucker geben. Das geht genauso wie mit dem Bildschirm, über einen PRINT-Befehl. Zum Unterschied, daß nicht der Bildschirm gemeint ist, steht hinter dem Befehl das Zeichen »#« (engl. Abkürzung für »Number«=Nummer), gefolgt von derselben File-Nummer, die beim OPEN-Befehl verwendet wurde. Dahinter steht, getrennt durch ein Komma, der auszudruckende String. Das alles sehen Sie in Zeile 278.

```
278 PRINT#1,T$;
```

Bitte vergessen Sie nicht, am Ende des PRINT#-Befehls (sprich: »Print-Number« oder »Print-Nummer«) ein Semikolon zu setzen.

Ohne dieses würde nach jedem Ausdruck einer Zeile ein doppelter Zeilenvorschub ausgeführt werden, da wir ja das Ausdrucken mit der RETURN-Taste auslösen, die ihrerseits einen Zeilenvorschub bewirkt.

Zum Beenden des Schreibens nehmen wir wieder dieselbe Methode wie bei unserem ersten Programm, die Abfrage der Tastenkombination CTRL-A mit ihrem ASCII-Code 1. Auch hier gilt für den VC 20 mein Vorschlag, die F1-Taste mit dem Codewert 133 zu nehmen. Sobald diese Tasten gedrückt sind, muß die Verbindung zum Drucker geschlossen werden. Das besorgt der CLOSE-Befehl, hinter dem lediglich die File-Nummer der Datei stehen muß, die eingangs geöffnet worden ist. Diese Prüfzeile sieht so aus:

```
246 IF ASC(A$)=1   THEN CLOSE 1:END
246 IF ASC(A$)=133 THEN CLOSE 1:END
(nur für den VC 20)
```

Da diese Zeile mitten im Programm steht, muß sie mit END abgeschlossen werden.

Wir merken uns:

1. Der OPEN-Befehl stellt eine Verbindung zu einem angeschlossenen Ein-/Ausgabegerät her. Man sagt auch, er öffnet eine Datei (File) zum Schreiben oder Lesen.
2. Hinter dem OPEN-Befehl stehen weitere Angaben, mindestens zwei, maximal sechs. Die drei wichtigsten sind: File-Nr., Geräte-Nr., Sekundäradresse
3. Der PRINT#-Befehl sendet Daten an ein mit OPEN angewähltes Gerät. Maximal können mit einem PRINT#-Befehl 255 Zeichen gesendet werden. Dem PRINT#-Befehl muß dieselbe File-Nummer folgen, wie dem OPEN-Befehl, der die Datei eröffnet hat.
4. Der CLOSE-Befehl schließt eine eröffnete Datei. Er wird ebenfalls mit der entsprechenden File-Nummer versehen.

Starten Sie jetzt einmal dieses Programmfragment mit RUN. Der Drucker druckt nach jedem eingetippten Buchstaben den jeweils letzten Stand der String-Addition T$ aus, allerdings immer in eine neue Zeile. Das müssen wir verhindern. Wir müssen ebenfalls erreichen, daß eine Zeile erst nach ihrer Fertigstellung ausgedruckt wird. Eine Zeile soll erst dann »fertig« sein, wenn die RETURN-Taste gedrückt wurde.

Diese Prüfung ist für uns einfach, haben wir sie doch schon früher angewendet. Diese Prüfung muß vor dem PRINT#-Befehl erfolgen, deshalb setzen wir sie zwischen die Zeile 264 und 278:

```
268 IF ASC(A$)<>13 THEN 230
280 T$=""
```

Zur Erinnerung: ASC(A$) bildet den ASCII-Code des Zeichens A$, 13 ist der Code der RETURN-Taste. Nach Drücken der RETURN-Taste ist also die Zeile zu Ende, und der String T$ wird komplett ausgedruckt. Bevor eine neue Zeile beginnt, muß die »alte« gelöscht werden. Das geschieht in Zeile 280, indem dem String T$ kein Wert - darge-

Druckerausgabe

stellt durch zwei Gänsefüßchen hintereinander - zugewiesen wird. Bis hierher steht also das folgende Teilprogramm:

```
200 REM --- LISTING ---
204 PRINT CHR$(14) CHR$(147)
206 OPEN 1,4,7
230 GET A$
234 PRINT"*{CRSR LINKS}";
236 FOR T=0 TO 30:NEXT T
238 PRINT" {CRSR LINKS}";
240 FOR T=0 TO 30:NEXT T
244 IF A$=""THEN 230
246 IF ASC(A$)=1 THEN CLOSE 1:END
(246 IF ASC(A$)=133 THEN CLOSE 1:END (nur VC 20)
262 T$=T$+A$
264 PRINT A$;
```

```
268 IF ASC(A$)<>13 THEN 230
278 PRINT#1,T$
280 T$=""
282 GOTO 230
```

Das Ergebnis kommt einer Schreibmaschine schon recht nahe. Lästig ist jedoch, daß wir sehr aufpassen müssen, nicht zu viele Zeichen in eine Zeile zu schreiben, bevor wir die RETURN-Taste drücken. Auch die Verwendung der Cursor-Tasten kann zum Steckenbleiben des Druckers führen. Da muß noch etwas geschehen.

Bei einer Schreibmaschine kann man sowohl den Abstand vom linken Papierrand einstellen als auch die Zeilenlänge. Unser Programm soll das natürlich auch können. In den folgenden Zeilen wird die maximale Zeilenlänge eingestellt und in Zeile 254 ausgeführt.

```
212 INPUT "ZEICHEN PRO ZEILE";Z
214 IF Z>80 THEN 212
254 IF LEN(T$)=Z THEN 230
```

Da die meisten Drucker nur 80 Zeichen pro Zeile (DIN-A4-Format) drucken können, darf die Variable Z nicht größer als 80 sein, was in Zeile 214 nachgeprüft wird. Zeile 254 prüft anschließend, ob die Länge des Strings T$ schon den vorgegebenen Wert von Z erreicht hat. Ist das der Fall, bleibt das Programm in der GET-Schleife der Zeilen 230 und 254 stehen. Leider kommen wir aus dieser Schleife auch mit der RETURN-Taste nicht heraus, da sie ja erst später in Zeile 268 abgefragt wird.

Wir müssen das in Zeile 254 berücksichtigen und die RETURN-Taste nach Erreichen der Grenze Z noch zulassen. Ich lasse die DEL-Taste, mit der wir Tippfehler korrigieren können, auch zu. Zeile 254 wird erweitert:

```
254 IF LEN(T$)=Z AND ASC(A$) <> 13
AND ASC(A$) <> 20 THEN 230
```


Gleich nach der Zeilenlänge legen wir den linken Randabstand L fest:

```
218 INPUT"LINKER RAND";L
```

Auch dieser Rand darf eine maximale Größe nicht überschreiten. Wäre er 80, dann hätten auf der Seite keine Zeichen mehr Platz. Er hängt also auch davon ab, wieviel Zeichen in der Zeile stehen sollen. Das heißt, daß die Summe der beiden Werte, L und Z, die Zahl 80 nicht überschreiten darf:

```
220 IF L+Z>79 THEN 218
```

So, wie es jetzt programmiert ist, bestimmt die Zeilenlänge Z die maximale Größe des linken Randes. Man kann natürlich auch zuerst den Rand festlegen, der seinerseits dann die maximale Zeilenlänge beeinflußt.

Interessant wird nun, wie der linke Rand in den Druckvorgang eingebaut wird. Ich halte es für eine gute Idee, nach dem Ende des Eintippens einer Zeile einfach so viele Leerstellen vor den String T$ zu setzen, wie durch den Wert L vorgegeben ist. Das darf natürlich erst nach dem Drücken der RETURN-Taste, muß aber noch vor dem Ausdrucken per PRINT#-Befehl erfolgen, also zwischen den Zeilen 268 und 278.

```
272 FOR J=1 TO L
274 T$=" "+T$
276 NEXT J
```

Sie sehen, die String-Addition leistet uns hier auch wieder gute Dienste.

Probieren Sie ruhig das bisher Erreichte aus. Wir nähern uns unserem Ziel langsam, aber sicher.

Die Tasten <CURSOR-UP>, <CURSOR-DOWN> und <CURSOR-LINKS> sind Störenfriede. Wir wollen sie deshalb unterdrücken, gleich nach der Eingabe des Zeichens (GET-Schleife) mit der schon bekannten ASC-Umwandlung und Abfrage. Die ASCII-Codes der drei betroffenen Cursor-Tasten entnehmen wir dem Handbuch, sie sind 157,145 und 17.

```
248 IF ASC(A$)=157 THEN 230
250 IF ASC(A$)=145 THEN 230
252 IF ASC(A$)= 17 THEN 230
```

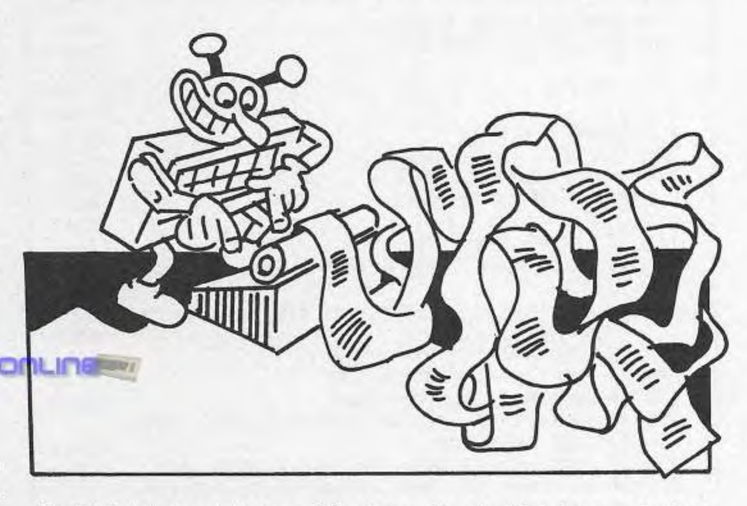
Als weitere Störer kommen die Farb-Tasten und die INST-Taste in Frage, aber deren Ausschluß will ich Ihnen überlassen; die Methode ist ja jetzt bekannt.

DEL bedeutet DELETE (engl.: streichen, löschen), das heißt, wir können mit der DEL-Taste in der getippten Zeile herumkorrigieren, bevor sie zum Drucken gegeben wird.

Dadurch wird allerdings die Länge des Strings T$ verkleinert – also auch die maximale Zeilenlänge. Außerdem darf es nicht möglich sein, mit der DEL-Taste über den linken Rand hinweg in die vorhergehende – schon abgeschlossene – Zeile zu gelangen. Das letztere verhindern wir mit Zeile 256; die Längenkorrektur besorgt Zeile 258:

```
256 IF ASC(A$)=20 AND LEN(T$)=0 THEN 230
258 IF ASC(A$)=20 THEN T$=LEFT$(T$,LEN(T$)-1):
    GOTO 264
```

Die Zeile 256 ist wohl leicht verständlich. Die Zeile 258, so fürchte ich, bedarf einer näheren Erklärung. Wenn wir



die DEL-Taste drücken, löschen wir ein Zeichen auf dem Bildschirm. Der String T$ verkürzt sich um ein Zeichen, aber nur auf dem Bildschirm. Damit die Längenprüfung der Zeile 254 nicht durcheinander gerät, müssen wir die aktuelle Länge von T$ um 1 verringern. Der Befehl LEFT$(T$,X) schneidet bekanntlich aus einem String T$ von links her genau X Zeichen heraus. In Zeile 258 habe ich für diesen Wert X die Länge von T$ minus 1 genommen, dargestellt durch den Ausdruck LEN(T$)-1.

Nach Zeile 258 folgt Zeile 262, durch deren Wirkung die als letztes Zeichen eingegebene DEL-Taste zum String T$ addiert werden würde. Um das zu verhindern, hängen wir in der Zeile 258 einen Sprungbefehl auf die übernächste Zeile 264 an.

Das Programm »SCHREIBMASCHINE« ist eigentlich komplett. Mir fallen lediglich drei Verbesserungen ein, die man einfügen könnte:

- Einstellung des Zeilenabstandes
- Klingel bei Zeilenende
- Ausdrucken des gesamten Textes

Den Zeilenabstand kann man bei fast allen Druckern verändern. Leider ist die Methode nicht bei allen Druckern gleich. Deshalb habe ich in diesem Programm darauf verzichtet. Falls Sie es aber für Ihren Drucker vorsehen wollen, gebe ich Ihnen einen Hinweis.

Die Abfrage des einzustellenden Wertes kann in einer Zeile 222 erfolgen, wieder mit INPUT. Darauf folgt eine Abfrage auf einen zulässigen Wert.

```
222 INPUT"ZEILENABSTAND";A
224 IF A < MIN OR A > MAX THEN 222
226 ...Befehl für Umschaltung...
```



In Zeile 224 sollten Sie die minimal und maximal zugelassenen Werte für den Zeilenabstand überprüfen. Die Werte selbst sowie den Befehl für die Umschaltung müssen Sie Ihrem Drucker-Handbuch entnehmen.

Das Ertönen einer Klingel einige Zeichen vor dem Zeilenende ist nicht schwer zu programmieren. Nur ist die Erzeugung von Tönen bei jedem der Commodore-Computer verschieden. Deshalb beschränke ich mich hier ebenfalls nur auf einen prinzipiellen Hinweis. Im kompletten Listing 14 ist eine Version für den C 64 angegeben.

Ich schlage vor, am Ende des Programms ab Zeile 286 ein Unterprogramm anzuhängen, welches angesprungen wird, sobald das fünftletzte Zeichen einer Zeile erreicht ist. Die Prüfung erfolgt kurz vor dem Rücksprung auf eine neue Zeicheneingabe:

```
266 IF LEN(T$)=Z-5 THEN GOSUB 286
286 ....Unterprogramm KLINGEL ...
299 RETURN
```

Das Unterprogramm besteht aus dem Auswählen einer Tonhöhe, der Lautstärke, dem Einschalten des Tones und, nach einer kurzen Zeitverzögerung, aus dem Ausschalten des Tones.

Ich möchte die Perfektionierung dieses Programms noch eine Stufe weiter treiben. Wir können bis jetzt nur zeilenweise ausdrucken.

Eine interessante Variante wäre die Möglichkeit, den ganzen geschriebenen Text auf einen Schlag auszudrucken. Dazu müssen wir ihn aber speichern. Wir müssen also die einzelnen Zeilen T$ genauso aneinanderhängen wie die einzelnen Zeichen A$. Das könnte so aussehen:

```
279 Z$=Z$+T$
```

Einer String-Variablen Z$ können wir aber nur maximal 255 Zeichen zuweisen. Das Zusammenbinden mehrerer Zeilen, die einzeln maximal 80 Zeichen enthalten können, würde aber sehr rasch, nämlich ungefähr nach vier Zeilen, zu einer verbotenen Stringlänge und damit zum Abbruch des Programms führen. Aber es gibt eine Lösung dafür.

Wir können jede einzelne Zeile T$ in einem eindimensionalen Feld (Array) abspeichern und bei Bedarf das gesamte Feld ausdrucken.

Für diejenigen Leser, denen das Arbeiten mit Feldern nicht ganz geläufig ist, will ich das Verfahren kurz erläutern:

Unser Ziel ist es, die einzelnen Zeilen, so wie sie mit dem Drücken der RETURN-Taste ausgedruckt werden, der Reihe nach zu speichern. Wir müssen dazu jede Zeile der Reihe nach numerieren und ihr den jeweiligen Text T$ zuordnen.

```
279 Z1$=T$
```

In Zeile 279 ist eine derartige Zuordnung angebracht. Aber nach jeder neuen Zeile müßte die Zahl hinter dem Z um 1 erhöht werden. Das geht aber nur, wenn wir Z$ als sogenannte Feld-Variable definieren und sie so schreiben:

```
Z$(1) =T$
```

Die 1 in der Klammer können wir durch eine normale Variable ersetzen und in einer Schleife nach jedem Durchgang hochzählen.

```
278 PRINT#1,T$;
279 Z$(X)=T$
280 T$=""
281 X=X+1
282 GOTO 230
```

Bei Beginn des Programms steht X auf Null, und die erste Zeile T$ wird der Feld-Variablen Z$(0) zugeordnet, die nächste Zeile der Feld-Variablen Z$(1) und so fort.

Wenn wir nur 11 solcher Feld-Variablen – also 11 Zeilen – haben, brauchen wir nichts weiter tun. Für mehr Zeilen müssen wir den nötigen Platz »reservieren«. Das besorgt der DIM-Befehl.

Wie groß wird wohl der notwendige Platz sein? Ein Text hat zirka 60 Zeilen pro Seite. Wenn wir maximal zehn Seiten schreiben wollen, dann muß das Feld 600 Plätze haben.

```
208 DIM Z$(600)
```

Natürlich können wir vorsorglich mehr Platz reservieren. Der Haken an der Sache ist, daß die Reservierung Speicherplatz belegt, und davon haben zum Beispiel der VC 20 und der C 16 ohne Speichererweiterung recht wenig zur Verfügung. Sie können das aber ruhig probieren. Wenn der Speicher nicht reicht, dann meldet dies der Computer sofort nach RUN mit einer entsprechenden Fehlermeldung. Sie müssen dann halt die Zahl hinter dem DIM-Befehl kleiner werden lassen.

```
200 REM *** LISTING SCHREIBMASCHINE ***
202 :                                          <178>
204 PRINT CHR$(14)CHR$(147)                    <205>
206 OPEN 1,4,7                                 <195>
208 DIM Z$(600)                                <042>
210 :                                          <186>
212 INPUT"ZEICHEN PRO ZEILE";Z                 <019>
214 IF Z>80 THEN 212                           <051>
216 :                                          <192>
218 INPUT"LINKER RAND";L                       <063>
220 IF L+Z>79 THEN 218                         <196>
228 :                                          <204>
230 GET A$                                     <184>
232 :                                          <208>
234 PRINT"*{LEFT}";                            <219>
236 FOR T=0 TO 30:NEXT                         <053>
238 PRINT"{SPACE,LEFT}";                       <080>
240 FOR T=0 TO 30:NEXT                         <057>
242 :                                          <218>
244 IF A$="" THEN 230                          <181>
246 IF ASC(A$)=1 THEN CLOSE 1:END:REM VC-2
    0 133                                      <225>
247 IF ASC(A$)=2 THEN GOSUB 310:REM VC-20
    134                                        <055>
248 IF ASC(A$)=157 THEN 230                    <232>
250 IF ASC(A$)=145 THEN 230                    <214>
252 IF ASC(A$)=17 THEN 230                     <105>
254 IF LEN(T$)=Z AND ASC(A$)<>13 AND ASC(A
    $)<>20 THEN 230                            <018>
256 IF ASC(A$)=20 AND LEN(T$)=0 THEN 230       <136>
258 IF ASC(A$)=20 THEN T$=LEFT$(T$,LEN(T$)
    -1):GOTO 264                               <000>
260 :                                          <238>
262 T$=T$+A$                                   <022>
264 PRINT A$;                                  <035>
266 IF LEN(T$)=Z-5 THEN GOSUB 286              <199>
268 IF ASC(A$)<>13 THEN 230                    <208>
270 :                                          <248>
272 FOR J=1 TO L                               <010>
274 T$=" "+T$                                  <110>
276 NEXT J                                     <114>
278 PRINT#1,T$;                                <245>
279 Z$(X)=T$                                   <111>
280 T$=""                                      <131>
281 X=X+1                                      <055>
282 GOTO 230                                   <012>
284 :                                          <006>
286 REM ++++ UNTERPROGRAMM KLINGEL ++++        <009>
287 REM          (BEISPIEL C-64)               <105>
288 :                                          <117>
289 POKE 54277,28                              <139>
290 POKE 54273,26                              <130>
292 POKE 54276,33                              <252>
294 POKE 54296,5                               <183>
296 FOR K=1 TO 100:NEXT K                      <220>
298 POKE 54276,32                              <254>
299 RETURN                                     <103>
300 :                                          <022>
310 REM++ UNTERPROGRAMM GANZER TEXT +++        <045>
312 :                                          <034>
314 FOR Y=0 TO X                               <146>
316 PRINT#1,Z$(Y);                             <234>
318 NEXT Y                                     <020>
320 RETURN                                     <124>
```

Listing 14. Die Verwertung unserer Kenntnisse über Strings ermöglicht ein kleines Textverarbeitungsprogramm: »Schreibmaschine«

Wir merken uns:
1. Der DIM-Befehl reserviert einen Speicherbereich für eine durch den dahinter in der Klammer stehenden Zahl festgelegte Anzahl von Variablen. Diese Variablen haben alle denselben Namen und unterscheiden sich nur durch eine fortlaufende Zahl, die ebenfalls in Klammern steht.
2. Der durch den DIM-Befehl reservierte Speicherbereich wird FELD oder ARRAY genannt. Entsprechend heißen die »numerierten« Variablen Feld- oder Array-Variable.
3. Mit DIM können sowohl Felder für numerische als auch für String-Variable reserviert werden. Ein Feld kann immer nur einen einzigen dieser Variablen-Typen enthalten.

Also, die zusätzlichen Befehle lauten:

```
208 DIM Z$(600)
247 IF ASC(A$)=2 THEN GOSUB 310
279 Z$(X)=T$
281 X=X+1
```

Ich habe eine noch nicht erwähnte Zeile 247 eingeschmuggelt. Sie steht bei den anderen Abfrage-Zeilen und prüft, ob eine Taste mit dem ASCII-Code 2 gedrückt worden ist. Dieser Code entspricht der Tastenkombination CTRL-B, mit der wir auf ein Unterprogramm ab Zeile 310 springen, welches den ganzen angesammelten Text – das heißt das ganze Feld – ausdruckt.

Für den VC 20 wählen wir die F3-Taste zur Auslösung dieses Sprunges. Sie hat den ASCII-Code 134:

```
247 IF ASC(A$)=134 THEN GOSUB 310  (nur für VC 20)
```

Das Unterprogramm ab Zeile 310 sieht so aus:

```
310 REM  UNTERPROGRAMM
314 FOR Y=0 TO X
316 PRINT#1,Z$(Y);
318 NEXT Y
320 RETURN
```

Sobald wir das Unterprogramm anspringen, enthält die Zählvariable X die Zahl der Zeilen, die eingegeben worden sind. In einer weiteren Schleife fangen wir eine neue Zählung an, und zwar von Null bis zum letzten Wert X. Die Zählvariable dafür nenne ich Y. In dieser Schleife werden alle Eintragungen des Feldes Z$(Y) der Reihe nach ausgedruckt.

Auf diese Weise überlisten wir die Einschränkung von Basic, nämlich nur 255 Zeichen in einer String-Variablen aneinanderreihen zu dürfen.

Im Listing 14 ist das vollständige Programm – ohne die Einstellung des Zeilenabstandes beim Drucker – im Zusammenhang ausgedruckt.

3. Formatieren von Text und Zahlen

Zahlen in Reih und Glied oder in Tabellen zu schreiben, ist auf der Schreibmaschine auch für eine geübte Schreibkraft nicht ganz einfach. Ihr hilft nur der sogenannte Tabulator oder das schlichte Auszählen der Anschläge. Der Computer kennt den Tabulator auch, aber das Auszählen übernimmt natürlich ein kleines Programm.

Linksbündige Formatierung

Die Funktion des Tabulators übernimmt der Basic-Befehl TAB(X). Er setzt eine nachfolgende Zahl an den mit X bezeichneten Platz. Wie das Beispiel

```
PRINT TAB(3) 22.43
```

zeigt, fängt der TAB-Befehl mit dem Zählen ab der Spalte 0 an, reserviert in der Spalte 3 einen Platz für das Vorzeichen und schreibt die Zahl 22.43 ab der vierten Spalte. Diese Eigenheit muß man kennen.

Wollen Sie beispielsweise ganze Zahlenkolonnen auf dem Bildschirm untereinander ausgeben, sollten Sie diese Eigenschaft bei der linksbündigen Formatierung auf jeden Fall berücksichtigen.

Dazu ein kurzes Programm-Beispiel:

```
20 REM----LINKSBUENDIG----
30 A=123.5
35 B=-2175.334
40 C=0.23
45 D=22.4567
60 PRINT TAB(12) A
65 PRINT TAB(12) B
70 PRINT TAB(12) C
75 PRINT TAB(12) D
```

Zuerst definiert man also die Werte der vier Variablen A bis D. Dann lassen sie sich mit TAB (12) ausdrucken.

An den oberen Bildschirmrand kann man noch eine Zahlenreihe gefolgt von einer Leerzeile schreiben, die das Abzählen der Stellen erleichtert:

```
50 PRINT"0123456789012345678901234567890"
55 PRINT
```

Nach <SHIFT-CLR/HOME> und RUN sieht man auf dem Bildschirm die in Bild 6 gezeigte Darstellung:

```
RUN
012345678901  234567890123456789

              123.5
             -2175.334
              .23
              22.4567
```

Bild 6. Linksbündige Formatierung mit dem TAB-Befehl. Beachten Sie die Position der Vorzeichen.

```
RUN
012345678901  234567890123456789

              123.5
             -2175.334
              .23
              22.4567
       123.5
   -2175.334
         .23
     22.4567
```

Bild 7. Neben der linksbündigen Formatierung gestattet der TAB-Befehl auch die rechtsbündige

Wie oben beschrieben, steht das negative Vorzeichen in der Spalte 12, alles andere ab Spalte 13. Und in für uns ungewohnter Art wird in der amerikanischen Schreibweise des Computers die Null vor dem Dezimalpunkt weggelassen. Daran muß man sich gewöhnen.

Rechtsbündige Formatierung mit TAB

Die rechtsbündige Ausrichtung von Zahlenkolonnen ist schon etwas aufwendiger, da sie ja von der Länge der Zahlen abhängt. Der Befehl TAB(12) gibt wie vorher die 12. Spalte als Fixpunkt aus, nur muß davon noch die Ziffernanzahl der Zahl abgezogen werden:

```
PRINT TAB(12-Ziffernzahl) Zahl
```

Zur Bestimmung der Ziffernzahl verwendet man den Befehl LEN(X$).

Sie sehen, dieser Befehl gilt nur für die Länge eines Strings. In der Anwendung muß man also die Zahlen erst in Strings umwandeln, bevor man ihre Länge feststellen kann. Auch dafür kennen wir einen Befehl. Er lautet, wie in folgendem Beispiel zu sehen, STR$(X).

Die hier folgenden Programmzeilen machen das für die vier Zahlenwerte A bis D:

```
100 REM----RECHTSBUENDIG MIT TAB---
110 A$=STR$(A)
115 B$=STR$(B)
120 C$=STR$(C)
125 D$=STR$(D)
```

Danach folgen die mit dem LEN-Befehl veränderten Druckbefehle:

```
130 PRINT TAB(12-LEN(A$)) A
135 PRINT TAB(12-LEN(B$)) B
```

```
140 PRINT TAB(12-LEN(C$)) C
145 PRINT TAB(12-LEN(D$)) D
150 PRINT
```

Nach Löschen des Bildschirms und RUN erhält man jetzt den in Bild 7 dargestellten erweiterten Ausdruck:

Zum Nachprüfen nehmen wir Zeile 130.
- LEN(A$) ergibt die Zahl 6 (der Vorzeichenplatz und der Dezimalpunkt zählen mit!).
- (12-LEN(A$)) ergibt demnach ebenfalls 6.
- Das heißt, daß die Zahl A ab der 7. Spalte beginnt.

Rechtsbündige Formatierung mit RIGHT$

Wie so oft beim Computer, gibt es auch hier mehrere Lösungen für ein Problem. Anstelle des TAB-Befehls läßt sich auch der String-Befehl RIGHT$ einsetzen. Dieser Befehl - mit der Schreibweise RIGHT$(X$,Z) - schneidet aus einem String X$ von rechts her Z Stellen heraus.
PRINT RIGHT$("ABEND",3) ergibt END.

Um zu sehen, was wir in unserem Beispiel machen müssen, schauen wir uns die Situation noch einmal genauer in Bild 8 an.

Mit dem Befehl

```
PRINT RIGHT$(A$,12)
```

könnte man von rechts her genau zwölf Stellen herausschneiden und ab dem linken Rand auf den Bildschirm schreiben, wenn A$ aus der Zahl 123.5 und mehr als zwölf Leerstellen davor (!) bestehen würde, also:

```
PRINT RIGHT$("     "+A$,12)
```

Nichts ist leichter als das. Man erzeugt einfach einen String Z$, der aus lauter Leerstellen - vorsichtshalber eine ganze Bildschirmzeile lang - besteht. Diesen String setzten wir vor die Zahlen.

```
200 REM----RECHTSBUENDIG MIT STRINGS-----
210 FOR I=1 TO 40
215 Z$=Z$+" "
220 NEXT
```

Diese drei Zeilen erzeugen durch 40faches Addieren von Leerstellen den gewünschten Leerstellen-String Z$.

```
225 PRINT RIGHT$(Z$+A$,12)
```

Die Zeile 225 bildet aus Z$ und A$ einen String aus 46 Zeichen, wobei die Zahl A ganz rechts steht. Mit RIGHT$ (Z$+A$,12) werden von rechts zwölf Zeichen herausgeschnitten und mit PRINT ausgedruckt.

Entsprechend lauten die restlichen Zeilen:

```
230 PRINT RIGHT$(Z$+B$,12)
235 PRINT RIGHT$(Z$+C$,12)
240 PRINT RIGHT$(Z$+D$,12)
```

Das Resultat ist in Bild 9 dargestellt.

Das haben wir gelernt: Formatierte Zahlen

1. Linksbündige Formatierung von Zahlen macht man mit TAB(X).
2. Zur rechtsbündigen Formatierung von Zahlen mit TAB(X) muß die Zahlenlänge vom Argument X des TAB-Befehls abgezogen werden.
3. Man kann auch mit dem String-Befehl RIGHT$ rechtsbündig formatieren, indem links vor der Zahl Leerstellen hinzugefügt werden und eine mit dem Argument X identische Anzahl von Stellen herausgeschnitten wird.

Die bisherigen drei Programmteile sind in Listing 15 zusammengefaßt dargestellt.

Dezimalpunkt-Formatierung mit TAB

Eine linksbündige oder rechtsbündige Formatierung von Zahlen mag sicher ihre Anwendung haben, gebräuchlich ist sie aber nicht. Wir sind eher gewohnt, alle Einer-, Zehner-, Hunderterstellen untereinander zu schreiben. Das ist die Formatierung mit dem Dezimalpunkt.

Dazu benötigt man einen Programmteil, der feststellt, wo innerhalb der Zahl der Dezimalpunkt liegt. Uns hilft dabei der Basic-Befehl MID$. In der Schreibweise MID$(K$,M,N) schneidet er aus dem String K$ von links ab der M-ten Stelle N Zeichen heraus.
PRINT MID$("RECHTS",2,4) ergibt ECHT (Bild 10).

Statt der vier Werte A bis D nimmt man jetzt eine Zahl K. Sie werden gleich sehen, warum. Sie wird wie vorher in einen String verwandelt (Listing 16):

```
340 K$=STR$(K)
```

Die nächsten drei Zeilen

```
345 FOR J=1 TO 40
350 V$=MID$(K$,J,1)
355 IF V$ < >"." THEN NEXT J
```

schneiden aus dem String K$ jeweils eine Ziffer heraus, und zwar in einer Schleife von der ersten bis zur 40. Ziffer des Strings K$. Bei jedem Schritt wird geprüft, ob das Resultat V$ ein Dezimalpunkt ist. Ist das nicht der Fall, geht die

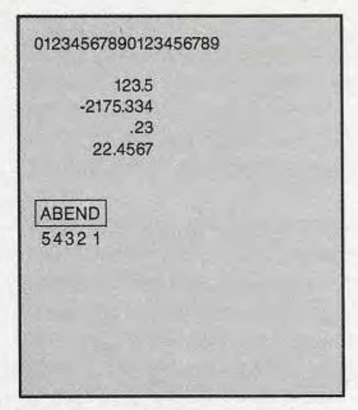


Bild 8. Das Prinzip der rechtsbündigen Formatierung mit dem RIGHT$-Befehl ist gut zu erkennen

```
RUN
012345678901234567890123456789
           123.5
           -2175.334
           .23
           22.4567
       123.5
   -2175.334
         .23
     22.4567
       123.5
   -2175.334
         .23
     22.4567
```

Bild 9. Rechtsbündiges Formatieren mit dem Right$-Befehl ist einfach durchzuführen

```
10 REM***FORMATIERTE ZAHL***                  <197>
15 :                                          <247>
20 REM----LINKSBUENDIG------                  <118>
25 :                                          <001>
30 A=123.5                                    <041>
35 B=-2175.334                                <225>
40 C=0.23                                     <183>
45 D=22.4567                                  <094>
50 PRINT"012345678901234567890123456789012
   3456789                                    <236>
55 PRINT                                      <157>
60 PRINT TAB(12) A                            <002>
65 PRINT TAB(12) B                            <136>
70 PRINT TAB(12) C                            <013>
75 PRINT TAB(12) D                            <147>
95 :                                          <071>
100 REM----RECHTSBUENDIG MIT TAB---           <029>
105 :                                         <081>
110 A$=STR$(A)                                <042>
115 B$=STR$(B)                                <180>
120 C$=STR$(C)                                <061>
125 D$=STR$(D)                                <199>
130 PRINT TAB(12-LEN(A$)) A                   <102>
135 PRINT TAB(12-LEN(B$)) B                   <141>
140 PRINT TAB(12-LEN(C$)) C                   <180>
145 PRINT TAB(12-LEN(D$)) D                   <219>
150 PRINT                                     <252>
190 :                                         <166>
200 REM----RECHTSBUENDIG MIT STRINGS----      <033>
205 :                                         <181>
210 FOR I=1 TO 40                             <230>
215 Z$=Z$+" "                                 <155>
220 NEXT                                      <230>
225 PRINT RIGHT$(Z$+A$,12)                    <202>
230 PRINT RIGHT$(Z$+B$,12)                    <208>
235 PRINT RIGHT$(Z$+C$,12)                    <214>
240 PRINT RIGHT$(Z$+D$,12)                    <220>
290 :                                         <012>
```

Listing 15. Formatierte Zahlenkolonnen

```
300 REM ****** DEZIMAL-FORMAT ********          <246>
305 :                                          <027>
307 :                                          <029>
310 REM---DEZIMAL-FORMAT MIT TAB---            <088>
315 :                                          <037>
320 PRINT"0123456789012345678901234567890123456789
                                               <252>
325 PRINT                                      <173>
330 READ K                                     <196>
335 IF K=0.0 THEN 400                          <253>
340 K$=STR$(K)                                 <063>
345 FOR J=1 TO 40                              <119>
350 V$=MID$(K$,J,1)                            <075>
355 IF V$<>"." THEN NEXT J                     <066>
360 PRINT TAB(12-J) K                          <020>
365 J=40                                       <118>
370 GOTO 330                                   <108>
375 DATA 123.5,-2175.334,0.23,22.4567,0.0      <223>
399 :                                          <121>
400 REM---DEZIMAL-FORMAT MIT STRINGS----       <082>
405 :                                          <127>
410 FOR I=1 TO 40                              <176>
415 Z$=Z$+" "                                  <101>
420 NEXT                                       <176>
425 PRINT                                      <017>
430 READ K                                     <040>
435 IF K=0.0 THEN 500                          <099>
440 K$=STR$(K)                                 <163>
445 FOR J=1 TO 40                              <219>
450 V$=MID$(K$,J,1)                            <175>
455 IF V$<>"." THEN NEXT J                     <166>
460 PRINT RIGHT$(Z$+K$,12+(LEN(K$)-J))         <084>
465 J=40                                       <218>
470 GOTO 430                                   <216>
475 DATA 123.5,-2175.334,0.23,22.4567,0.0      <067>
490 :                                          <212>
500 REM----ZAHLEN GLEICHER LAENGE-----         <165>
505 :                                          <227>
510 FOR I=1 TO 8                               <230>
515 Z$=Z$+" "                                  <203>
520 NEXT                                       <022>
523 L=7                                        <072>
525 PRINT                                      <119>
530 READ K                                     <142>
535 IF K=0.0 THEN END                          <031>
540 K1$=STR$(K)                                <080>
545 K2$=MID$(K1$,1,L)                          <030>
560 PRINT TAB(12) RIGHT$(Z$+K2$,L)             <182>
570 GOTO 530                                   <070>
575 DATA 123.5,-2175.334,0.23,22.4567,0.0      <169>
```

Listing 16. Dezimalpunkt-Formatierung mit dem TAB-Befehl als Beispielprogramm

Schleife einen Schritt weiter. Der Rücksprung von Zeile 355 erfolgt mit NEXT J, in Zeile 345 wird nach einem ergebnislosen Vergleich der Schleifenzähler hochgezählt.

Ist es der Dezimalpunkt, bleibt die Zählvariable J auf ihrem Wert stehen, der damit der Anzahl der Ziffern links vom Dezimalpunkt entspricht. Diese Zahl J muß vom TAB-Wert 12 abgezogen werden, damit die Zahl K mit ihrem Dezimalpunkt auf die 12. Spalte geschrieben wird:

```
360 PRINT TAB(12-J) K
```

Die vier Zahlen werden diesmal in einer DATA-Zeile gespeichert und mit dem READ-Befehl ausgelesen. Deshalb braucht man dafür, wie oben erwähnt, nur eine Variable:

```
330 READ K
335 IF K=0.0 THEN 400
365 J=40
370 GOTO 330
375 DATA 123.5,-2175.334,0.23,22.4567,0.0
```

Nach dem Ausdrucken der ersten Zahl muß die Zählvariable auf ihren Endwert gesetzt werden (Zeile 365), bevor die nächste Zahl durch den Rücksprung-Befehl in Zeile 370 gelesen werden kann. Nach jedem READ-Befehl wird in Zeile 335 geprüft, ob die »Endmarkierung« in der DATA-Zeile erreicht ist. Diese Markierung muß eine Zahl sein, die selbst als Ausdruck nicht vorkommen kann. In diesem Fall wurde die Zahl 0.0 gewählt. Tritt sie auf, dann springt das



Bild 10. So funktioniert der MID$-Befehl: aus »RECHTS«wird »ECHT«

```
RUN
012345678901 234567890123456789

        123 5
      -2175 334
            23
         22 4567
```

Bild 11. Dezimalpunkt-Formatierung mit dem TAB-Befehl

Programm auf den nächsten, in unserem Beispiel noch nicht vorhandenen Programmteil ab Zeile 400.

Um das Programm zu vervollständigen, übernehmen wir aus dem Listing 15 die Zeilen 50 und 55 mit der Ziffernleiste, hier aber mit den Zeilennummern 320 und 325:

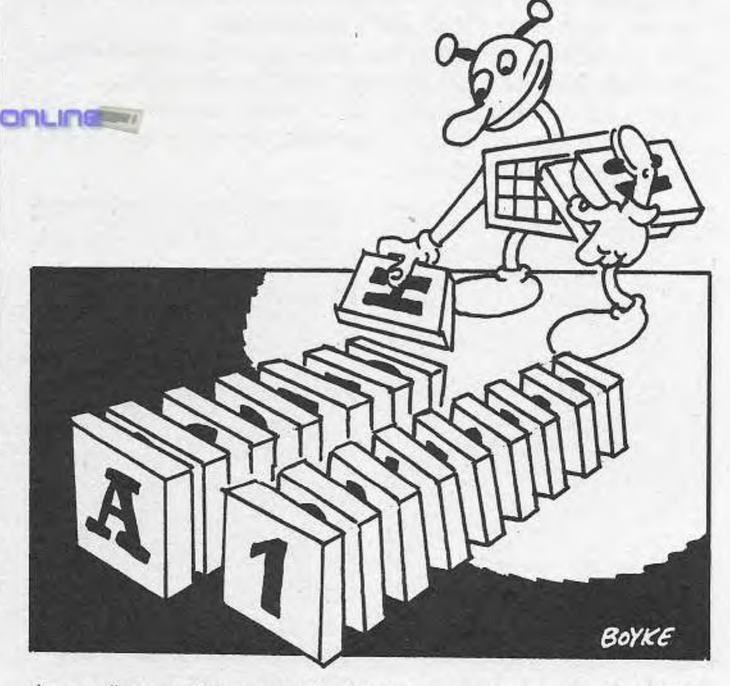
```
320 PRINT "0123456789012345678901234567890123456789"
325 PRINT
```

Das ganze Programm ergibt als Resultat Bild 11.

Genau wie im obigen Beispiel des rechtsbündigen Formatierens kann die Dezimalpunkt-Formatierung statt mit dem TAB-Befehl auch mit dem RIGHT$-Befehl realisiert werden.

Der einzige Unterschied zu dem TAB-Verfahren liegt darin, daß auch hier der Zahlen-String K$ mit dem Leerstellen-String Z$ verlängert wird.

Nur die Berechnung, wie viele Zeichen von rechts her aus dem überlangen String Z$+K$ herausgeschniten werden müssen, ist etwas komplizierter. Schauen Sie bitte noch mal Bild 8 an. Statt jeweils zwölf Zeichen abzuschneiden und auszudrucken, muß man jetzt 12 + Anzahl der Ziffern rechts vom Dezimalpunkt abschneiden. Diese Zahl ist aber leicht errechenbar aus der Länge der Zahl minus der Anzahl der Ziffern links vom Dezimalpunkt. Wir kennen sie schon, es ist der Stand der Zählvariable J.



Der Print-Befehl lautet jetzt:

```
PRINT RIGHT$(Z$+K$,12+(LEN(K$)-J))
```

Dieser Programmteil kann sehr leicht durch Überschreiben der 300er Zeilennummern – natürlich nur dort, wo man die Befehle übernimmt – erzeugt werden. Er sieht so aus:

```
400 REM---DEZIMAL-FORMAT MIT STRINGS-------
410 FOR I=1 TO 40
415 Z$=Z$+" "
420 NEXT
```

```
425 PRINT
430 READ K
435 IF K=0.0 THEN 500
440 K$=STR$(K)
445 FOR J=1 TO 40
450 V$=MID$(K$,J,1)
455 IF V$ < > "." THEN NEXT J
460 PRINT RIGHT$(Z$+K$,12+(LEN(K$)-J))
465 J=40
470 GOTO 430
475 DATA 123.5,-2175.334,0.23,22.4567,0.0
```

Sie sehen, alle Teile außer Zeile 460 kennen Sie schon. Das Resultat ist identisch mit dem der Zeilen 300 bis 375.

Das haben wir gelernt: Dezimalpunkt-Formatierung

1. Zahlen können auf den Dezimalpunkt ausgerichtet untereinander geschrieben werden, indem mit dem MID$-Befehl der Platz des Dezimalpunktes ermittelt und auf ihn rechtsbündig geschrieben wird.
2. Die Methoden zur Rechtsbündigkeit sind die gleichen wie vorher.

Formatierte Zahlen gleicher Länge

Bislang wurden Zahlen verwendet, die verschieden lang sind, sowohl vor als auch hinter dem Dezimalpunkt. In vielen Fällen, in denen Zahlen in Tabellen – meistens mit beschränktem Platz – geschrieben werden, müssen sie die gleiche Länge haben. Die Methoden, dies zu erreichen, sind die gleichen wie die bisher gezeigten. Ein Beispiel soll es verdeutlichen. Ich gebe folgende Regeln vor:

- Die Zahlen sind wieder der Variablen K zugeordnet und werden aus einer DATA-Zeile ausgelesen.
- Die Variable L bestimmt die zugelassene Gesamtlänge der Zahl (inklusive Vorzeichen und Dezimalpunkt).
- Wenn Zahlen länger sind als L, dann werden sie von rechts her abgeschnitten, das heißt, sie verlieren Stellen hinter dem Dezimalpunkt.

```
RUN
0123456789012345678901234567890123456789

          123.5
        -2175.334
              .23
           22.4567

          123.5
        -2175.334
              .23
           22.4567

            123.5
          -2175.3
              .23
           22.456
```

Bild 12. Wie man im Beispiel sieht, werden ab der 13. Spalte die Zahlen rechtsbündig ausgegeben

```
RUN
012345678901|234567890123456789
            |LEITERWAGEN
            |HAUS
            |TAB-BEFEHL
            |ZEICHEN
 LEITERWAGEN|
        HAUS|
  TAB-BEFEHL|
     ZEICHEN|
 LEITERWAGEN|
        HAUS|
  TAB-BEFEHL|
     ZEICHEN|
```

Bild 13. Rechts- und linksbündige Texte werden ausgegeben entsprechend der Formatierung

Wir können die meisten Teile des vorherigen Programms ab Zeile 400 verwenden:

```
500 REM---ZAHLEN GLEICHER LAENGE----
510 FOR I=1 TO 8
515 Z$=Z$+" "
520 NEXT
523 L=7
525 PRINT
530 READ K
535 IF K=0.0 THEN END
540 K1$=STR$(K)
570 GOTO 530
575 DATA 123.5,-2175.334,0.23,22.4567,0.0
```

Bis hierher kennen Sie schon alles.

Es wurde allerdings eine neue Zeile 523 eingeschoben, welche die gewünschte Länge der Zahlen vorgibt. Und in Zeile 540, in der die Zahl K in einen String umgewandelt wird, taucht jetzt als String-Variable K1$ auf. Man braucht nämlich noch eine zweite String-Variable für K, bei der die Zahlen rechts auf die Länge L gestutzt werden. Das macht man wieder mit dem MID$-Befehl:

```
545 K2$=MID$(K1$,1,L)
```

Diese Zeile schneidet aus K1$ ab der ersten Ziffer von links her L Ziffern heraus.

Wir könnten natürlich die Zeilen 540 und 545 zusammenziehen und so K1$ und K2$ sparen:

```
K$=MID$(STR$(K),1,L)
```

Aber das ist nicht so verständlich.

Die Links- und Rechtsbündigkeit erzielen wir wie gehabt:

```
560 PRINT TAB(12) RIGHT$(Z$+K2$,L)
```

Mit RUN 500 und durch Verändern des Wertes von L können Sie die Wirkungsweise ausprobieren.

Diese elf Zeilen können, wie vorher bei K$ schon angedeutet, zusammengeschoben werden. Die Methode führt allerdings zu der oft beklagten Unleserlichkeit von String-Befehlen. Zur Demonstration:

```
500 REM---
530 READ K
535 IF K=0.0 THEN END
560 PRINT TAB(12) RIGHT$(" ")+MID$(STR$ (K),1,7),7)
570 GOTO 530
575 DATA        etc.
```

Es ist wie in der Mathematik. In Zeile 560 werden einfach alle Variablen eingesetzt. Das ist zwar kurz, aber übersichtlich ist es nicht.

Mit RUN erhalten Sie den in Bild 12 gezeigten Bildschirmausdruck.

Es steht Ihnen nun frei, alle genannten Methoden beliebig zu kombinieren.

Das haben wir gelernt: Formatierte Zahlen gleicher Länge

Um verschiedene Zahlen auf gleiche Länge zu bringen, werden sie mit dem MID$-Befehl von rechts her auf diese Länge abgeschnitten und dann wie oben mit RIGHT$ rechtsbündig auf dem Bildschirm oder Drucker ausgegeben.

Formatieren auf dem Drucker

An dieser Stelle zeigen wir Ihnen, warum die Formatierung mit String-Befehlen derjenigen mit dem TAB-Befehl vorzuziehen ist.

Wenn Sie eine Zahlenreihe formatiert auf Ihrem Drucker ausgeben wollen, gibt es mit dem TAB-Befehl je nach Druckertyp manchmal Schwierigkeiten, die mit einem gezielten Einsatz von String-Befehlen von vornherein vermieden werden können.

Um das im Programm einmal auszuprobieren, müssen Sie im obigen Programmteil (ab Zeile 400) am Anfang den Drucker anwählen und ihm die »Befehlsgewalt« geben. Nach erfolgreicher Prüfung auf 0.0 ist die Verbindung wieder zu unterbrechen.

```
406 OPEN 1,4:CMD 1
430 IF K=0.0 THEN PRINT#1:CLOSE 1: END
```

Den PRINT-Befehl in Zeile 460 brauchen Sie nicht wie sonst nach einem OPEN-Befehl in einen entsprechenden

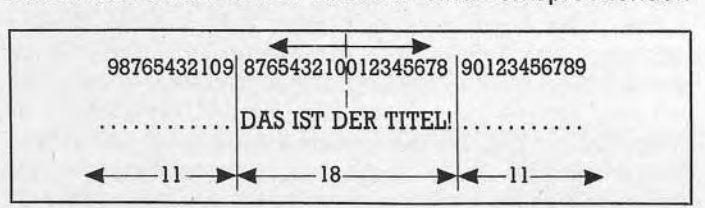


Bild 14. So funktioniert das Zentrieren von Texten

```
10 REM******* LISTING 3/3 ***********           <192>
15 :                                            <247>
20 REM-----ZENTRIERTE TEXTE ------              <095>
25 :                                            <001>
30 PRINT"98765432109876543210012345678901 2
   3456789"                                     <031>
35 PRINT                                        <137>
40 READ A$                                      <146>
45 IF A$="@" THEN END                           <024>
50 PRINT TAB((40-LEN(A$))/2) A$                 <077>
55 GOTO 40                                      <001>
70 DATA DAS IST DER TITEL!,UEBERSCHRIFT,S
   P I E L B E G I N N                          <008>
75 DATA BASIC-KURS,STRING-ECKE,"123456",@       <142>
```

Listing 17. Das Demo druckt mehrere in einer DATA-Zeile abgelegte Überschriften zentriert aus

PRINT#-Befehl umzuwandeln, weil in Zeile 406 ja der CMD-Befehl verwendet wird – ein Vorteil, der leider viel zu selten beachtet wird.

Das haben wir gelernt: Formatierte Ausgabe auf dem Drucker

1. Zum Formatieren auf dem Drucker sind die Methoden mit String-Befehlen denjenigen mit dem TAB-Befehl vorzuziehen.
2. Wenn nach der Druckeranwahl mit OPEN 1,4 der CMD-Befehl genommen wird, können alle »normalen« PRINT-Befehle wie beim Bildschirm auch verwendet werden.

Formatierte Texte

Das linksbündige und rechtsbündige Formatieren von Texten geht im Prinzip mit der gleichen Methode wie bei den Zahlen. Das ist auch leicht verständlich, da wir ja die Zahlen in Strings umgewandelt und sie dann wie Texte weiterverarbeitet haben.

Links- und rechtsbündige Texte

Listing 15 läßt sich sehr leicht auf Text umstellen. Statt der Zahlen A, B, C und D gibt man Strings ein:

```
30 A$="LEITERWAGEN"
35 B$="HAUS"
40 C$="TAB-BEFEHL"
50 D$="ZEICHEN"
```

Entsprechend müssen in den Zeilen 60 bis 75 und 130 bis 145 die Zahlen A bis D in Stringvariable A$ bis D$ umgewandelt werden. Zu guter Letzt entfallen die Zeilen 110 bis 125, da eine weitere Umwandlung in Strings nicht mehr notwendig ist.

Dieses abgewandelte Programm liefert den in Bild 13 dargestellten Ausdruck.

Wie zu erwarten, werden die definierten Textvariablen rechts- und linksbündig ausgegeben.

Das ist so einfach, daß das ganze, geänderte Programm hier nicht wiedergegeben wird.

Das haben wir gelernt: Links- und rechtsbündige Texte

Das rechts- und linksbündige Formatieren von Texten macht man mit denselben Methoden wie bei den Zahlen.

Zentrieren

Unter Zentrieren versteht man das Anordnen eines Textes genau in der Mitte des Bildschirms oder des Papiers. Die häufigste Anwendung dürfte sicher die Überschrift sein.

Um zu sehen, wie das funktioniert, nehmen wir als Beispiel die Überschrift
»DAS IST DER TITEL!«.

Diese Überschrift ist 18 Zeichen lang. Da der Bildschirm aus 40 Spalten besteht, bleiben uns 40-18=22 Leerstellen, die gleichmäßig links und rechts von der Überschrift anzuordnen sind. Bild 14 zeigt, wie das funktioniert.

Die Ziffernleiste ist so aufgebaut, daß sie mit zwei Nullen in der Mitte seitensymmetrisch ist. So kann man die Zentrierung leicht nachprüfen. Sie sehen, was wir gemacht haben. Um das zu verstehen, eine Übersicht:

– wir haben die Überschrift definiert	A$= "DAS IST DER TITEL!"
– wir haben die Länge davon genommen	L=LEN(A$)
– wir haben die Länge von der Spaltenzahl des Bildschirms abgezogen	D=40-L
– wir haben die Differenz halbiert	H=D/2
– wir haben mit diesem Wert als TAB-Argument die Überschrift ausgedruckt	PRINT TAB(H) A$

In einem Demonstrationsprogramm (Listing 17) wird gezeigt, wie man mehrere in einer Datenzeile abgelegte Überschriften zentriert ausdruckt:

Dieses Programm erzeugt den in Bild 15 dargestellten Bildschirmausdruck.

Bei den Texten mit ungerader Zeichenzahl ist die Zentrierung um eine Stelle verrückt. Auch Zahlen kann man zentrieren, wenn sie als Strings eingegeben sind, wie die unterste Zeile in Bild 15 zeigt.

Das haben wir gelernt: Text zentrieren

Zum Zentrieren von Strings wird ihre Länge bestimmt und von der Bildschirm- oder Papierbreite abgezogen. Der Rest wird halbiert. Das daraus resultierende Ergebis wird als Maß für Linksbündigkeit genommen.

4. Laufschrift und Farbenspiele

Ein hübsches Programm beginnt meistens auch mit einem hübschen Vorspann, der oft aus bewegten Titeln besteht, die schnell Aufmerksamkeit erregen.

```
RUN
9876543210987654321001234567890123456789

           DAS IST DER TITEL!
             UEBERSCHRIFT
           S P I E L B E G I N N
              BASIC-KURS
              STRING-ECKE
                123456
```

Bild 15. Ausdruck des Demoprogramms, in dem Überschriften zentriert auf dem Bildschirm oder Drucker erscheinen (siehe Zahlenreihe oben)

Titel können in mehreren Arten bewegt werden – von links nach rechts, von rechts nach links, in Einzelbuchstaben oder als ganzer Block. Überall haben die Strings ihre Hand im Spiel.

Textbaustein von links

Zuerst soll ein Titel von links nach rechts mit einzelnen Buchstaben aufgebaut werden. Dazu picken wir mit dem

MID$-Befehl die Buchstaben der Reihe nach aus dem Titel und drucken sie auf den Bildschirm.

```
10 REM*****         *****
30 A$="DAS IST DER TITEL!"
35 S=1
40 T=1
45 PRINT MID$(A$,S,T);
```

Die entscheidende Zeile ist Zeile 45. In ihr wird mit dem MID$-Befehl jeweils nur ein Zeichen (T=1) des Textes A$ ab dem S-ten Buchstaben herausgepickt, wobei am Anfang S den Wert 1 hat. Wenn S nun in einer Zählschleife laufend um 1 erhöht wird, erscheint schrittweise der nächste Buchstabe von der linken Seite her. Das Semikolon hinter dem PRINT-Befehl unterdrückt den Zeilenumbruch.

Die Schleife bilden wir mit:

```
60 S=S+1
70 GOTO 45
```

Das Ende der Schleife stellen wir in Zeile 65 fest – mit der Frage, ob S die Gesamtlänge des Strings A$ schon erreicht hat. Es hilft dabei der LEN-Befehl.

```
65 IF S >LEN(A$) THEN END
```

Alles in Bewegung

Damit der ganze Vorgang nicht zu schnell abläuft, ist eine Verzögerung einzufügen:

```
50 FOR Z=1 TO 100: NEXT Z
```

Diese Schleife zählt zwischen jedem Ausdrucken eines Buchstabens von 1 bis 100, nicht in Sekunden, sondern in viel kürzeren Abständen. Mit der Wahl der oberen Grenze von Z können Sie die Geschwindigkeit festlegen.

Hereinschieben von links

Schwieriger wird es schon, wenn der String nicht aus einzelnen Zeichen aufgebaut wird, sondern sich in den Bildschirm hineinschieben soll. Das bedeutet nämlich, daß er rückwärts gedruckt werden muß.

Die meisten Zeilen bleiben im Prinzip identisch, auch der PRINT-Befehl. Bei MID$ muß mit dem letzten Buchstaben angefangen werden. S beginnt also bei einem Wert, den wir mit S=LEN(A$) errechnen. Dann wird S in der Schleife immer um 1 verringert, bis Null erreicht ist.

```
130 A$="DAS IST DER TITEL!"
135 S=LEN(A$)
140 T=1
160 S=S-1
165 IF S=0 THEN END
```

Damit der Text lesbar bleibt, darf man nicht jeweils nur einen Buchstaben ausdrucken (T=1), sondern zuerst einen, dann zwei, dann drei und so weiter. Das heißt, T muß immer um 1 hochgezählt werden.

```
155 T=T+1
```

Der Rücksprung und die Zeitschleife können unverändert übernommen werden:

```
150 FOR Z=1 TO 100:NEXT Z
170 GOTO 145
```

Jetzt fehlt nur noch der PRINT-Befehl. Wenn man ihn so schreibt wie vorher, nämlich:

```
145 PRINT MID$(A$,S,T);
```

also mit Semikolon, dann schiebt sich zwar der Text von links herein, aber nicht schrittweise. Versuchen Sie es ruhig einmal.

Vielmehr ist der jeweilige Text durch Weglassen des Semikolons immer wieder neu zu schreiben. Damit er in der gleichen Zeile bleibt, muß hinter dem PRINT-Befehl der Befehl CRSR-UP stehen. Wir erreichen das mit dem Befehl CHR$(145) – 145 ist der Code für CRSR-UP.

```
145 PRINT CHR$(145) MID$(A$,S,T)
```

Wenn der Befehlsteil CHR$(145) fehlt, werden die Einzelteile des Titels untereinander geschrieben, weil das in Zeile 45 verwendete Semikolon fehlt. Probieren Sie das mal aus.

Mit CRSR-UP wird jedes neue Titelfragment, das eigentlich in eine neue Zeile gedruckt wird, in die alte Zeile gehoben, wo es das vorhergehende Fragment überschreibt. Dadurch entsteht der Eindruck, daß der Text immer länger und von links in den Bildschirm hereingeschoben wird, allerdings nur solange, bis er komplett vorhanden ist.

Nach links hinausschieben

Wir wollen jetzt die Richtung umdrehen und den Text von rechts nach links aus dem Bildschirm hinausschieben.

```
230 A$="DAS IST DER TITEL!"
245 PRINT CHR$(145) MID$(A$,S,T)
250 FOR Z=1 TO 100:NEXT Z
270 GOTO 245
```

Wir fangen jetzt mit dem Text in voller Länge an

```
240 T=LEN(A$)
```

und zwar ab dem ersten Zeichen

```
235 S=1
265 IF S >LEN(A$) THEN END
```

Wenn wir S hochzählen, T aber konstant lassen,

```
260 S=S+1
```

dann wird im zweiten Schritt der volle Text ohne das erste Zeichen an die Stelle des alten Textes geschrieben, und zwar so lange, bis die Prüfung in Zeile 265 ergibt, daß der Wert von S die volle Textlänge erreicht hat.

Der Titel wandert somit aus dem Bildschirm heraus.

Nur eins ist noch unschön. Der letzte Buchstabe des Textes bleibt stehen, da er durch das um ein Zeichen kürzere neue Fragment nicht überschrieben werden kann. Wir können das korrigieren, indem wir am Schluß des Textes noch ein Leerzeichen einfügen. Dieses, wenn es stehen bleibt, ist ja nicht sichtbar.

```
230 A$="DAS IST DER TITEL! "
```

Von rechts nach links durchschieben

Da der Text so rasch verschwindet, soll er diesmal quer über den ganzen Bildschirm wandern und am linken Rand wieder verschwinden, so daß er rechts herein- und links hinausgeschoben wird.

Wie machen wir das?

Der erste Trick besteht darin, in die Zeilen 310 bis 330 vor den Text mindestens 39 Leerstellen zu schreiben, so daß

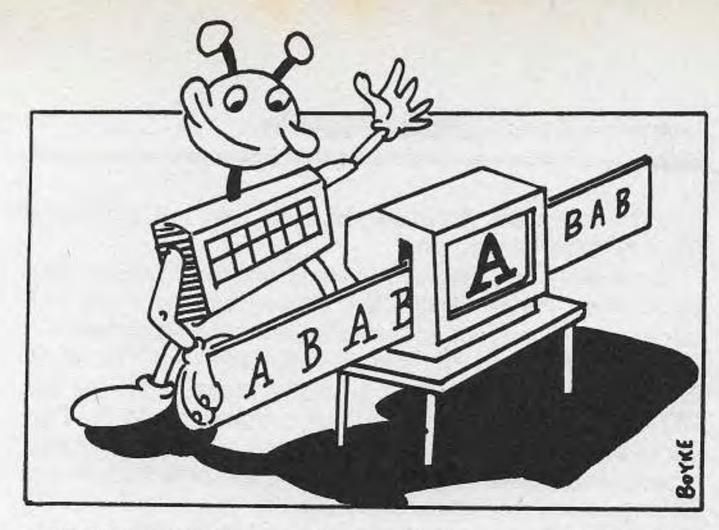


der eigentliche Titel erst ab dem 40. Zeichen des Strings A$ beginnt.

```
310 FOR X=1 TO 40
315 B$=B$+" "
320 NEXT X
325 A$="DAS IST DER TITEL!"
330 A$=B$+A$
```

Der zweite Trick betrifft den Wert von T. In Zeile 340 geben wir ihm nämlich den Wert 39.

Die unveränderte Zeile 345 schneidet jetzt aus dem überlangen String immer 39 Zeichen heraus, zuerst ab dem ersten Zeichen, danach ab dem zweiten und so fort. Dadurch



erscheinen zuerst nur die Leerstellen, dann Leerstellen plus dem ersten Buchstaben des Textes und so weiter, bis der Wert von S in Zeile 365 die Textlänge erreicht hat.

```
335 S=1
340 T=39
345 PRINT CHR$(145) MID$(A$,S,T)
350 FOR Z=1 TO 100:NEXT Z
360 S=S+1
365 IF S >LEN(A$) THEN END
370 GOTO 345
```

Wenn Sie T=40 wählen, um ganz am rechten Rand zu beginnen, wird der String zu lang, und es kommen einige Unstimmigkeiten im Ausdruck vor.

Eine letzte Variante verleiht der Laufschrift ein langes Leben. Wenn in der Zeile 365 das Ende der Schleife erreicht ist, soll das Programm nicht beendet werden, sondern mit Rücksetzung von S auf den Anfangswert 1 von neuem beginnen.

```
365 IF S >LEN(A$) THEN S=1
```

Die Laufschrift läßt sich durch <RUN/STOP> bremsen.

Von links nach rechts durchschieben

Der Text soll jetzt von links nach rechts laufen. Das Prinzip wurde schon erklärt, allerdings nur soweit, bis der ganze Text auf dem Bildschirm steht. Mit neuen Zeilennummern kann man das Programm direkt übernehmen:

```
400 REM----- nach rechts durchschieben ----
425 A$="DAS IST DER TITEL!"
435 S=LEN(A$)
440 T=1
445 PRINT CHR$(145) MID$(A$,S,T)
450 FOR Z=1 TO 100:NEXT Z
455 T=T+1
460 S=S-1
465 IF S=0 THEN END
470 GOTO 44
```

Der Trick für das Durchlaufen ist auch schon bekannt: Verlängern des Strings A$ mit Leerzeichen. Auch hier müssen die Leerzeichen links vor dem Text stehen:

```
410 FOR X=1 TO 40
415 B$=B$+" "
420 NEXT X
430 A$=B$+A$
```

Jetzt besteht der String A$ aus dem Text und Leerstellen auf der linken Seite.

Wie vorher läuft der Text von links nach rechts. Aber am rechten Rand angekommen, gerät er außer Rand und Band. Dadurch, daß er über die 40. Spalte hinaus geschrieben werden muß, erscheinen die Buchstaben natürlich in der nächsten Zeile.

Wie können wir erreichen, daß der Text buchstabenweise einfach hinter dem rechten Rand verschwindet?

Nun, sobald der String A$ des MID$-Befehls am rechten Bildschirmrand angelangt ist, ist er gerade 40 Zeichen lang (T=40). Nun soll der MID$-Befehl nicht die durch T hochgezählte Länge herausschneiden, sondern immer ein Zeichen weniger, damit es nicht in die nächste Zeile rutschen kann.

Wir definieren ab T größer 40 den Wert von T einfach neu:

```
464 IF T>40 THEN T=T-1
```

Nach dem Rücksprung zur Zeile 445 ist der Wert für T innerhalb des MID$-Befehls um eins kleiner geworden und verkürzt den Text auf der rechten Seite.

Um den Text zum Dauerläufer zu machen, verwenden wir in Zeile 465 nicht END, sondern einen Rücksprung zum Anfang:

```
465 IF S=0 THEN 425.
```

Das haben wir gelernt: Wie programmiert man Laufschrift?

1. Laufschrift kann man mit dem Befehl MID$(A$,S,T) erzeugen, indem man S und T variabel hält.
2. Für eine Bewegung von links nach rechts gilt:
   Anfangswerte S=LEN(A$)
   T=1
   S wird rückwärts, T aber vorwärts gezählt
3. Für eine Bewegung von rechts nach links gilt:
   Anfangswerte S=1 T=LEN(A$) oder T=39
   S wird vorwärts gezählt, T bleibt konstant
4. Um den Text durchlaufen zu lassen, wird er links durch String-Addition mit Leerstellen aufgefüllt.

Text in die Mitte schieben

Durch die Kombination des Zentrierens mit der durchlaufenden Schrift erhalten wir eine Laufschrift, die sich nur bis zur zentralen Lage in die Bildschirmmitte bewegt. Wir nehmen den Programmteil, der den Text von links nach rechts schiebt, und versehen ihn mit neuen Zeilennummern (ab 500).

Mit der Formel, die für die Mitte des Bildschirms entwickelt wurde (Zentrieren), müssen wir jetzt den Lauf der Schrift stoppen, wenn sie die Bildmitte erreicht hat. Man macht das mit einer Prüfung der Zählvariablen T in Zeile 564. Die Formel lautete:

40-LEN(A$)/2

Man steht aber zwei kleinen Problemen gegenüber:

Erstens gilt für LEN(A$) die ursprüngliche Länge des Textes, der in Zeile 525 definiert wurde. In Zeile 564 aber ist A$ bereits mit den vielen Leerstellen versehen. Wir müssen daher zwischen diesen beiden Strings unterscheiden, indem wir ihnen verschiedene Namen geben.

```
510 FOR X=1 TO 40
515 B$=B$+" "
520 NEXT X
525 C$="DAS IST DER TITEL!"
530 A$=B$+C$
535 S=LEN(A$)
540 T=1
545 PRINT CHR$(145) MID$(A$,S,T)
550 FOR Z=1 TO 100:NEXT
555 T=T+1
560 S=S-1
```

Die geänderten Zeilen sind 525 und 530, wobei der Text jetzt den Namen C$ hat.

Das zweite kleine Problem liegt darin, daß mit der Zentrierformel linksbündig zentriert wurde, während der MID$-Befehl der Zeile 545 auf der rechten Seite agiert. Die Lösung finden Sie bestimmt.

In Bild 16 sind die Formeln für die Längen der einzelnen Abschnitte eingetragen. Der unterste Teil läßt sich leicht ausrechnen:

(40-LEN(C$))/2 + 2*LEN(C$)/2 = (40+LEN(C$))/2

Auch durch Nachdenken kann man darauf kommen, daß man die Länge des Strings C$ nicht von der Bildschirmbreite abziehen, sondern mit ihr addieren und dann wie vorher

halbieren muß. Mit welcher Methode die Prüfzeile auch immer entwickelt wurde, sie schaut jetzt so aus:

```
564 IF T >(40+LEN(C$))/2 THEN END
570 GOTO 545
```

Die Prüfung auf S=0 kann man sich sparen.

Alle Programmteile der verschiedenen Laufschriften sind im Listing 18 zusammengefaßt.

Bei diesem Anwendungsbeispiel von String-Befehlen geht es um die Farbe des Bildschirm-Hintergrundes. Bekanntlich haben die Commodore-Computer mehrere Möglichkeiten, unterschiedliche Farben für den Bildschirm zu verwenden. Die einfachsten davon sind

- Zeichen-Farbe, Bildrand-Farbe, Hintergrund-Farbe.

Allgemein bekannt - weil in den Handbüchern erwähnt - ist die Methode, die Farbe des Hintergrundes und des Bildschirmrandes zu verändern. Dafür gibt es bestimmte Adressen im Speicher, in die entsprechende Zahlencodes hineingePOKEt werden müssen. Diese Adressen sind in der folgenden Tabelle zusammengefaßt:

C64/C128	
Umrandung	53280
Bildschirm	53281

Farbenspiele

Durch Ändern der Adressen können aber nur einfarbige Flächen erzeugt werden.

Wir wollen den Bildschirm-Hintergrund mit vielen verschiedenen Farben füllen. Wir verwenden dafür das Verfahren des »Reversen (oder invertierten) Leerzeichens«:

Wenn Sie zuerst die Tastenkombination <CTRL RVS/ON> und dann eine der Farbtasten zusammen mit der CTRL- oder der Commodore-Taste drücken, blinkt der Cursor in der neuen Farbe. Wird jetzt die Leertaste gedrückt, erzeugen Sie einen farbigen Streifen auf dem Bildschirm. Solange man im reversen Modus bleibt - also nicht <CTRL RVS/OFF> drückt - kann man beliebige Farben auswählen und Muster malen.

Das ist nichts Aufregendes. Jetzt aber soll das Ganze per Programm geschehen.

Zu Beginn werden alle Farben als String definiert und in Zeile 25 der Stringvariablen X$ zugeordnet.

```
25 X$="{BLK WHT RED CYN PUR GRN BLUYEL}"
```

Die einzelnen Farben werden bekanntlich durch Drücken der CTRL-Taste gleichzeitig mit der jeweiligen Farbtaste innerhalb von Gänsefüßchen erzeugt. Wenn das geschieht, erscheint ein für jede Farbe spezielles reverses Zeichen.

```
10 REM *** LAUFSCHRIFT***
15 :                                          <247>
20 REM---- TEXTAUFBAU VON LINKS-------        <059>
25 :                                          <001>
30 A$="DAS IST DER TITEL!"                    <160>
35 S=1                                        <024>
40 T=1                                        <033>
45 PRINT MID$(A$,S,T);                        <170>
50 FOR Z=1 TO 100:NEXT Z                      <077>
55 REM (T BLEIBT KONSTANT)                    <162>
60 S=S+1                                      <244>
65 IF S>LEN(A$) THEN END                      <094>
70 GOTO 45                                    <096>
95 :                                          <071>
100 REM----HEREINSCHIEBEN VON LINKS----       <130>
105 :                                         <081>
130 A$="DAS IST DER TITEL!"                   <004>
135 S=LEN(A$)                                 <053>
140 T=1                                       <133>
145 PRINT CHR$(145) MID$(A$,S,T)              <067>
150 FOR Z=1 TO 100: NEXT                      <012>
155 T=T+1                                     <103>
160 S=S-1                                     <120>
165 IF S=0 THEN END                           <199>
170 GOTO 145                                  <066>
190 :                                         <166>
200 REM--- NACH LINKS HINAUSSCHIEBEN ---      <067>
215 :                                         <191>
230 A$="DAS IST DER TITEL! "                  <104>
235 S=1                                       <224>
240 T=LEN(A$)                                 <162>
245 PRINT CHR$(145) MID$(A$,S,T)              <167>
250 FOR Z=1 TO 100: NEXT                      <112>
255 REM (T BLEIBT KONSTANT)                   <106>
260 S=S+1                                     <190>
265 IF S>LEN(A$)THEN END                      <040>
270 GOTO 245                                  <176>
290 :                                         <012>
300 REM---- NACH LINKS DURCHSCHIEBEN ---      <131>
305 :                                         <027>
310 FOR X=1 TO 40                             <196>
315 B$=B$+" "                                 <158>
320 NEXT X                                    <014>
325 A$="DAS IST DER TITEL! "                  <201>
330 A$=B$+A$                                  <204>
335 S=1                                       <070>
340 T=39                                      <150>
345 PRINT CHR$(145) MID$(A$,S,T)              <013>
350 FOR Z=1 TO 100: NEXT                      <214>
355 REM (T BLEIBT KONSTANT)                   <208>
360 S=S+1                                     <034>
365 IF S>LEN(A$) THEN S=1                     <115>
370 GOTO 345                                  <028>
390 :                                         <112>
400 REM---NACH RECHTS DURCHSCHIEBEN-----      <078>
405 :                                         <127>
410 FOR X=1 TO 40                             <040>
415 B$=B$+" "                                 <002>
420 NEXT X                                    <114>
425 A$="DAS IST DER TITEL!"                   <045>
430 A$=B$+A$                                  <048>
435 S=LEN(A$)                                 <099>
440 T=1                                       <179>
445 PRINT CHR$(145) MID$(A$,S,T)              <113>
450 FOR Z=1 TO 100:NEXT                       <058>
455 T=T+1                                     <149>
460 S=S-1                                     <166>
464 IF T>39 THEN T=T-1                        <190>
465 IF S=0 THEN 425                           <107>
470 GOTO 445                                  <136>
490 :                                         <212>
500 REM--- VON LINKS ZUR MITTE ---            <217>
505 :                                         <227>
510 FOR X=1 TO 40                             <140>
515 B$=B$+" "                                 <104>
520 NEXT X                                    <216>
525 C$="DAS IST DER TITEL!"                   <155>
530 A$=B$+C$                                  <152>
535 S=LEN(A$)                                 <201>
540 T=1                                       <025>
545 PRINT CHR$(145) MID$(A$,S,T)              <215>
550 FOR Z=1 TO 100:NEXT                       <160>
555 T=T+1                                     <251>
560 S=S-1                                     <012>
564 IF T>(40+LEN(C$))/2 THEN END              <066>
570 GOTO 545                                  <246>
590 :                                         <058>
600 REM--- VON RECHTS ZUR MITTE ---           <155>
605 :                                         <073>
610 FOR X=1 TO 40                             <242>
615 B$=B$+" "                                 <204>
620 NEXT X                                    <060>
625 C$="DAS IST DER TITEL! "                  <255>
630 A$=B$+C$                                  <252>
635 S=1                                       <116>
640 T=39                                      <196>
645 PRINT CHR$(145) MID$(A$,S,T)              <059>
650 FOR Z=1 TO 100:NEXT                       <004>
655 REM (TBLEIBT KONSTANT)                    <006>
660 S=S+1                                     <080>
664 IF S>(40+LEN(C$))/2 THEN END              <158>
670 GOTO 645                                  <098>
```

Listing 18. Laufschriften mit normalen Basic-Befehlen

In der nun folgenden Zeile 45 setzen wir wieder den MID$-Befehl ein, um jeweils eines davon herauszuschneiden und damit eine reverse Leerstelle zu drucken.

```
45 PRINT MID$(X$,Z,1) "{RVS-ON}";
```

Ab der Zahl Z wird vom String X$ ein Zeichen genommen. Wenn man zum Beispiel für Z die Zahl 5 wählt, ergibt das eine Leerstelle in der Farbe Purpur.

Vorsicht! Die Zahl Z hat nichts mit dem normalen Farbcode zu tun. Sie stellt lediglich die Position der entsprechenden Farbe im String X$ der Zeile 25 dar.

Spaßeshalber überlassen wir jetzt die Auswahl der Zahl Z dem Zufall mit der Zeile 35:

```
35 Z=INT(RND(1)*8)+1
```

Diese Zeile erzeugt eine Zufallszahl zwischen 1 und 8. Ein Rücksprung auf Zeile 35 und das Löschen des Bildschirms ganz am Anfang (in Zeile 20 das Semikolon nicht vergessen!) beschließt den ersten Teil unseres kleinen Programms:

```
10 REM----- FARBSPIEL ------
20 PRINT CHR$(147);
25 X$="{BLK WHT RED CYN PUR  GRN BLU YEL}"
35 Z=INT(RND(1)*8)+1
45 PRINT MID$(X$,Z,1) "{RVS-ON}";
55 GOTO 35
```

Nach RUN wählt die Zeile 35 immer neue Farbsymbole aus, die dann, durch das Semikolon am Ende des PRINT-Befehls aneinandergereiht, ausgedruckt werden.

Wir füllen dadurch den Bildschirm mit Farbklecksen, die zufallsbedingt verteilt werden.

Wenn wir noch eine Zählschleife einführen, können wir mehrere Zeichen der gleichen Farbe drucken:

```
40 FOR X=1 TO 40
50 NEXT X
```

Die Zahl 40 druckt jeweils eine ganze Zeile in einer Farbe, die Zahl 20 nur eine halbe Zeile.

Sie können sich so Ihren eigenen Farbeffekt heraussuchen.

Wir wollen jetzt die Farben nicht dauernd laufen lassen, sondern nur einen ganzen Bildschirm füllen. Wir verwenden dazu noch eine weitere Zählschleife, die außerhalb der »Zeichenschleife« (Zeilen 40 und 50) steht und 25 Zeilen nach unten zählt. Damit wir einen ungewünschten Zeilenhochschub am Ende vermeiden, zählen wir nur bis 24.

```
30 FOR K=1 TO 24
55 NEXT K
```

Diese Schleife ersetzt den Rücksprung mit GOTO.

Das haben wir gelernt: Programmieren farbiger Bildschirme

1. Auch Farben können mit String-Befehlen verarbeitet werden. Dazu müssen die Farben zuerst in einem String untergebracht werden.
2. Farben haben die gleichen Eigenschaften wie andere Zeichen und Buchstaben, wenn ihre reversen Symbole innerhalb von Gänsefüßchen stehen.

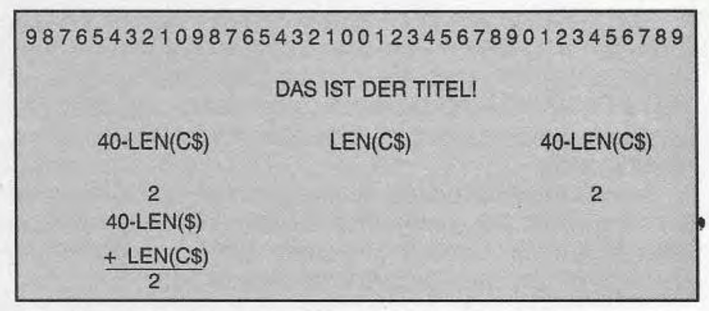


Bild 16. Der Text lernt laufen: Er wird mit Hilfe der Formel in die Mitte des Bildschirms geschoben

Vor einem derartigen bunten Hintergrund könnte jetzt die Laufschrift vorbeiwandern. Man muß dabei nur in Kauf nehmen, daß die Laufschrift in der aktuellen Zeile die Farben löscht und auf der ursprünglichen Hintergrundfarbe erscheint.

Wir nehmen dazu die Zeilen 410 bis 470 aus dem Listing 18. Jetzt wäre es schön, wenn die Commodore-Computer den MERGE-Befehl kennen würden, um die beiden Listings miteinander zu verknüpfen.

Das komplette Programm ist in Listing 19 wiedergegeben. Es enthält allerdings eine Version mit 16 Farben (Zeile 25). Außerdem druckt es keine Farbzeilen, sondern lauter Flecken (Zeile 30, Zeilen 40 und 50 sind durch REMs außer Funktion gesetzt).

5. Spielereien mit Wörtern und Texten

Welche Arbeiten sind eigentlich beim Sortieren erforderlich - wenn man die Kinder der Größe nach aufstellt, die Socken ihrer Farbe entsprechend zusammenlegt, die Namen alphabetisch einordnet und so weiter?

Prinzipiell nehmen wir eines der Objekte und vergleichen es mit seinen Nachbarn. Das ist in Bild 17 für die drei verschieden großen Männchen »Adi, Luis und Zenzi« dargestellt, die wir der Größe nach aufstellen wollen und daher erst einmal sortieren müssen.

Wir fangen willkürlich mit Luis an. Der Vergleich mit Zenzi ergibt, daß ihre Positionen stimmen, also machen wir nichts. Als nächstes vergleichen wir Zenzi mit Adi. Der Vergleich sagt uns, daß die beiden ihre Plätze tauschen müssen. Das geht aber nur, wenn einer der beiden so lange auf einen »Ausweichplatz« geht, bis der andere den neuen Platz eingenommen hat. Der dritte Vergleich, zwischen Adi und Luis, verlangt auch einen Platztausch.

Als Anwendungsbeispiel für das Sortieren von Strings bietet sich das alphabetische Ordnen von Namen an.

Wie vergleichen wir Namen?

```
10 REM ***** FARBENSPIELE ******
15 :                                         <247>
20 PRINT CHR$(147);                          <167>
25 X$="{BLACK,RED,CYAN,PURPLE,GREEN,BLUE,Y
   ELLOW,ORANGE,BROWN,LIG.RED,GREY 1,GREY
   2,LIG.GREEN,LIG.BLUE,GREY 3}"             <076>
30 FOR K=1 TO 24*40                          <239>
35 Z=INT(15*RND(0))+1                        <164>
40 REM FOR X=1 TO 2                          <049>
45 PRINT MID$(X$,Z,1)"{RVSON,SPACE}";        <221>
50 REM NEXT X                                <130>
55 NEXT K                                    <155>
60 PRINT"{RVOFF,WHITE}";                     <161>
65 :                                         <041>
410 FOR X=1 TO 40                            <040>
415 B$=B$+" "                                <002>
420 NEXT X                                   <114>
425 A$="DAS IST DER TITEL! "                 <045>
430 A$=B$+A$                                 <048>
435 S=LEN(A$)                                <099>
440 T=1                                      <179>
445 PRINT "{HOME}" TAB(240) TAB(240) CHR$(
    145) MID$(A$,S,T)                        <109>
450 FOR Z=1 TO 100:NEXT                      <058>
455 T=T+1                                    <149>
460 S=S-1                                    <166>
464 IF T>40 THEN T=T-1                       <156>
465 IF S=0 THEN 425                          <107>
470 GOTO 445                                 <136>
```

Listing 19. Farbenspielereien mit dem Computer



Das habe ich Ihnen im ersten Teil des Kurses erklärt, nämlich im Abschnitt »Welcher String ist größer oder kleiner«. Da die Vergleichsfunktionen größer, kleiner (>, <) die ASCII-Werte der Buchstaben vergleichen, liegt fest, daß Adi kleiner ist als Luis. Das A hat den ASCII-Wert 65, das L den Wert 76.

Zuerst wollen wir eine Liste anlegen, welche Familiennamen enthalten soll, zusammen mit den ersten Buchstaben des Vornamens, um bei möglicher Namensgleichheit ein weiteres Unterscheidungsmerkmal zu haben.

Bäumchen wechsle dich

Es gibt mehrere Möglichkeiten, eine Namensliste zu speichern. Ich wähle hier die Methode, die Namen aus DATA-Zeilen mit dem READ-Befehl in ein Feld (Array) einzulesen. Das hat den Vorteil, daß jeder Name mit seiner Position im Feld numeriert ist.

Ich habe heute keine Möglichkeit, auf die Eigenschaften der Felder näher einzugehen. Wenn Sie Einzelheiten darüber wissen wollen, finden Sie sie im 64'er-Sonderheft 40 im Basic-Kurs »Schritt für Schritt« in den Lektionen 19 und 20.

```
110 DATAKLINGE G,MAYER H, HÜBNER A,SCHRAMM K,
    HAUCK H
115 DATAMAYER D,WÄNGLER A, RÖDER T,ABSMEIER A,
    SCHNEIDER B,@
120 J=J+1
125 READ A$(J)
130 IF A$(J) <>"@" THEN 120
```

Die drei Zeilen 120 bis 130 lesen die Eintragungen der DATA-Zeilen in das Feld A$(J).

Ich habe mit Absicht keine FOR-NEXT-Schleife gewählt, weil wir dazu immer die genaue Anzahl der Namen wissen müssen. Mit obiger Zählschleife ist diese Anzahl unwichtig, solange die letzte Eintragung in den DATA-Zeilen ein »Stoppzeichen« ist. In unserem Beispiel ist es der Klammeraffe »@«.

Durch Zeile 130 wird der READ-Befehl so lange wiederholt, bis der Klammeraffe auftritt.

Bevor aber Daten in ein Feld hineingeschrieben werden, müssen wir seine Größe mit dem DIM-Befehl definieren. Um eine längere Liste zu ermöglichen, wähle ich als Argument die Zahl 20, das ergibt 21 Plätze (0 bis 20).

```
115 DIM A$(20)
```

Nach dem Einlesevorgang hat die Zählvariable J einen Wert erreicht, der um 1 höher ist als die Anzahl der Namen – weil ja der Klammeraffe dabei ist. Das wird weiter unten noch wichtig sein.

Die ausgewählte Sortiermethode ist sehr einfach. Die Namen werden der Reihe nach miteinander verglichen. Wenn von zwei benachbarten Namen A$(K) und A$(K+1) der erste »kleiner« (das heißt im Alphabet vor dem zweiten liegt) oder gleich ist, werden die nächsten beiden Namen A$ (K+1) und A$(K+2) miteinander verglichen.

```
135 FOR K=1 TO J-2
140 IF A$(K) <= A$(K+1) THEN 180
180 NEXT K
185 END
```

Ist die Bedingung der Zeile 140 nicht erfüllt, dann müssen die beiden Namen miteinander ihren Platz tauschen. Es ist aber, wie wir in Bild 1 gesehen haben, ein Ausweichplatz notwendig. Dazu wird in Zeile 145 eine zweite Stringvariable B$ eingeführt, über die der Ringtausch stattfindet. Das ist:

```
145   B$=A$(K)
150       A$(K)=A$(K+1)
155               A$(K+1)=B$
```

Ich hoffe, die Schreibweise dieser drei Zeilen macht den Vorgang deutlich.

Eigentlich könnten wir jetzt mit der K-Schleife in Zeile 135 fortfahren. Aber es kann ja sein, daß der vorgerückte Name noch weiter nach vorn kommen muß. Wir müssen daher die Zählvariable K um 1 zurücknehmen und den Vergleich der Zeile 140 wiederholen.

```
175   K=K-1:GOTO 140
```

Erst wenn dieser Rückwärtsgang keine Verschiebung mehr bringt, springt das Programm aus Zeile 140 nach 180, und die K-Schleife läuft weiter.

Bei einer bereits sortierten Liste würde die K-Schleife ungehindert durchlaufen, die Frage ist nur, wie oft.

Nun, wir haben oben gesehen, daß in der Liste, genauer gesagt im Feld A$(J), insgesamt J-1 Namen enthalten sind. Die Schleife braucht aber nur J-2mal durchlaufen werden, da im vorletzten Schritt bereits der letzte Name mit dem vorletzten verglichen wird.

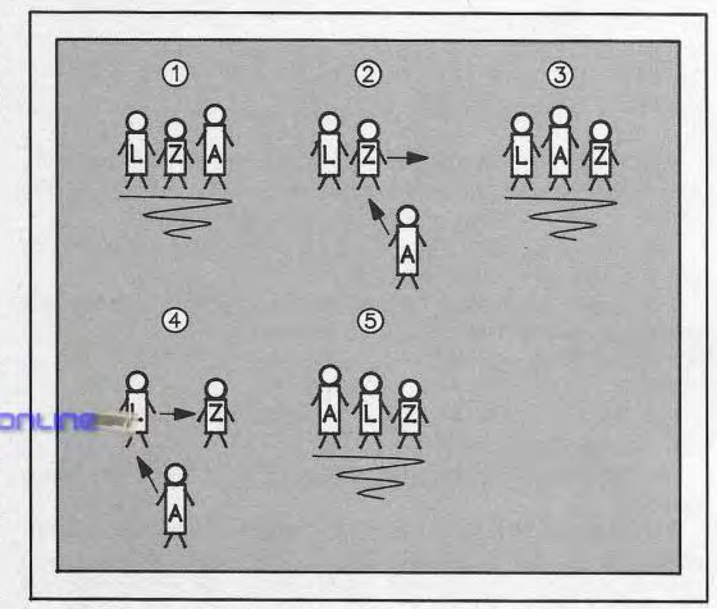


Bild 17. So wird einfach sortiert: Durch Vergleich der Elemente entsteht die richtige Reihenfolge.

Damit Sie in aller Ruhe sehen können, wie das Plätzetauschen vor sich geht, habe ich noch drei weitere Zeilen vorgesehen:

```
160 FOR I=1 TO J-1:PRINT A$(I)
165 NEXT I:PRINT
170 GET G$:IF G$=""THEN 170
```

Zuerst wird die gesamte Namensliste in ihrer derzeitigen Anordnung A$(I) ausgedruckt, gefolgt von einer Leerzeile.

Zeile 170 erlaubt Ihnen, das Resultat anzuschauen und zu vergleichen. Erst mit dem Druck irgendeiner Taste geht das Programm weiter. C 16- und C 128-Besitzer verwenden in Zeile 170 natürlich den praktischen GETKEY-Befehl:

```
170 GETKEY G$
```

Die für unsere String-Manipulationen wichtigsten Zeilen sind die Zeilen 140 bis 155.

Wortanalyse

Das ganze Programm ist in Listing 20 zusammengefaßt. Ich gebe zu, daß die hier verwendete Sortiermethode recht primitiv ist und viele Schritte braucht. Aber sie ist für Sie ein gutes Beispiel, wie ein Programm entsteht.

Neben Sortieren ist das Durchsuchen, Vergleichen und Analysieren von Texten ein klassisches Anwendungsgebiet der hohen Kunst, mit String-Befehlen zu programmieren.

```
100 REM***** SORTIEREN 1 *****
103 :                                           <079>
105 DATA KLINGE G,MAYER H,HUEBNER A,SCHRAM
    M K,HAUCK H                                 <087>
110 DATA MAYER D,WAENGLER A,ROEDER T,ABSME
    IER A,SCHNEIDER B,@                         <019>
112 :                                           <088>
115 DIM A$(20)                                  <171>
117 :                                           <093>
120 J=J+1                                       <123>
125 READ A$(J)                                  <018>
130 IF A$(J)<>"@" THEN 120                      <114>
132 :                                           <108>
135 FOR K=1 TO J-2                              <149>
140   IF A$(K) <= A$(K+1) THEN 180              <220>
143 :                                           <119>
145   B$=A$(K)                                  <200>
150      A$(K)=A$(K+1)                          <231>
155            A$(K+1)=B$                       <166>
157 :                                           <133>
160    FOR I=1 TO J-1:PRINT A$(I)               <191>
165    NEXT I:PRINT                             <207>
167 :                                           <143>
170   GET G$:IF G$="" THEN 170                  <031>
175   K=K-1:GOTO 140                            <063>
180 NEXT K                                      <024>
185 END                                         <187>
```

Listing 20. Der erste Schritt: die leicht zu verstehende Sortiermethode.

Was denken Sie, wenn Sie in einem Programm die folgenden Zeilen sehen?

```
135 XST$=MID$(VST$,M,LEN (NST$))
140 FOR L=1 TO LEN(NST$)
145 IF MID$(NST$,L,1)=MID$ (XST$,L,1) THEN X=X+1
```

Diese Zeilen entstammen einem später folgenden Listing, das ich natürlich Schritt für Schritt mit Ihnen durcharbeiten werde.

Als Aufgabe wähle ich ein kleines, fast triviales Programm, welches bei Angabe einer Stadt deren Postleitzahl sucht.

Zuerst legen wir eine Liste von Städten und Postleitzahlen an, und zwar in DATA-Zeilen, jeweils eine Postleitzahl und die entsprechende Stadt hintereinander, durch Kommata getrennt.

```
1000 DATA 8470,NABBURG, 5441,NACHTSHEIM
1005 DATA 6509,NACK,6506, NACKENHEIM
1010 DATA 8590,NAGEL,7270, NAGOLD,2061,NAHE
1015 DATA 2121,NAHRENDORF, 8674,NAILA
2000 DATA @,@
```

Zeile 2000 enthält zwei Endekennzeichen (Klammeraffe). Die Liste ist so aufgebaut, daß sie leicht verlängert und erweitert werden kann. Um aber jederzeit zu wissen, wie viele Einträge die Liste enthält, soll zu Beginn des Programms ein Teil stehen, der diese Anzahl bestimmt.

```
20 Z=0
25 READ PLZ$,ST$
```

Die Anzahl der Einträge wird mit der Variablen Z gezählt. Natürlich muß sie am Anfang auf 0 gesetzt sein. Dann lesen wir aus den DATA-Zeilen das erste Paar der Postleitzahlen und Städte (PLZ$ und ST$).

```
30 IF PLZ$ = "@" THEN 45
```

Zeile 30 prüft, ob wir das mit den Klammeraffen »@« markierte Ende der Liste schon erreicht haben. Wenn nicht, dann wird Z um 1 erhöht und das nächste Paar eingelesen.

```
35 Z=Z+1
40 GOTO 25
```

Der Rücksprung in Zeile 40 setzt die Zählung fort. Wird das Ende erkannt, dann springt Zeile 30 weiter auf Zeile 45, und da wollen wir vorläufig das Ergebnis der Zählung ausdrucken:

```
45 PRINT Z
```

Dieses Programm-Fragment mit RUN gestartet ergibt als Resultat die Zahl 9 – entsprechend der Zahl der eingetragenen Datenpaare.

Jetzt kommen die Datenpaare noch einmal an die Reihe, denn für den Suchvorgang wollen wir sie in ein Feld schreiben, dessen Größe wir mit dem DIM-Befehl festlegen müssen. Die Zahl Z gibt uns die Information dafür. Wir schreiben die Zeile 45 neu:

```
45 DIM PLZ$(Z),ST$(Z)
55 FOR I=1 TO Z
60 READ PLZ$(IZ),ST$(I)
65 NEXT I
```

Für die auf dem Gebiet der Felder (Arrays) Ungeübten sei hier erwähnt, daß es erlaubt ist, sowohl hinter dem DIM-Befehl mehrere Felder zu definieren als auch hinter dem READ-Befehl mehrere Datentypen einzulesen. Wir haben in den Zeilen 25, 45 und 60 beides für Postleitzahl und Stadt angewendet.

Die Zeilen 45 bis 65 besorgen das Einlesen der Datenpaare, diesmal mit einer FOR-NEXT-Schleife, da uns die Anzahl der notwendigen Schleifendurchgänge mit der Zahl Z ja bekannt ist.

Aber Vorsicht: Dadurch kommt in Zeile 60 der READ-Befehl zum zweiten Mal vor, so wie in Zeile 25. Das dürfen wir nicht so ohne weiteres. Zuerst muß ein interner Zähler, der die Reihenfolge beim READ-Befehl überwacht, auf seine Anfangsstellung gebracht werden. Das besorgt der RESTORE-Befehl, den wir in Zeile 50 einfügen.

```
50 RESTORE
```

Jetzt folgt der eigentliche Suchvorgang, der darin besteht, daß ein per INPUT-Befehl eingegebener Städtename mit allen Namen in der Liste verglichen wird. Ist er gefunden, wird er zusammen mit seiner Postleitzahl ausgedruckt, wenn nicht, dann meldet das Programm dieses Manko.

```
75 INPUT "NAME DER STADT";NST$
```

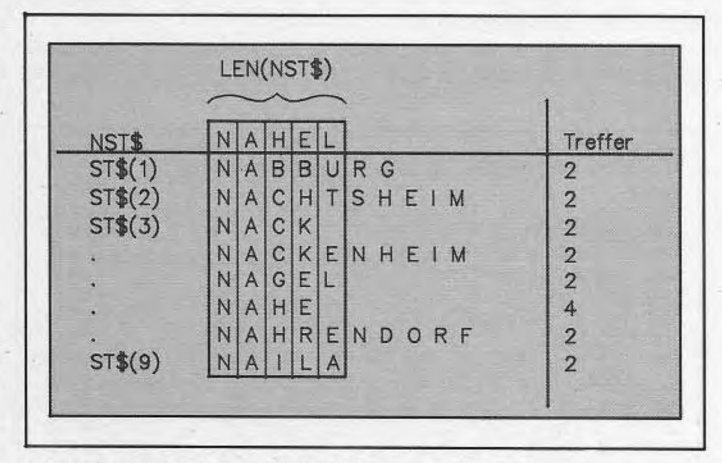


Bild 18. Vergleich durch Verkürzen der Worte

Für den einzugebenden Namen der Stadt müssen wir natürlich eine neue Variable angeben – NST$ (für Neue Stadt).

```
85 FOR K=1 TO Z
90 IF NST$ = ST$(K) THEN PRINT PLZ$(K),ST$(K):
   GOTO 75
95 NEXT K
110 PRINT "STADT NICHT IN DER LISTE"
```

Diese Suche durchläuft mit der K-Schleife das ganze Feld, also maximal Z mal. Wenn in Zeile 90 eine Übereinstimmung zwischen dem eingegebenen Städtenamen NST$ und dem jeweiligen gespeicherten Namen ST$(K) gefunden worden ist, dann wird Postleitzahl und Stadt ausgedruckt und eine neue Eingabe verlangt. Ist am Ende der

Schleife keine Übereinstimmung gefunden, meldet dies der PRINT-Befehl der Zeile 110.

Dieses Programm ist in sich geschlossen und könnte für eine eigene Postleitzahlenliste verwendet werden.

Abkürzung für Fehler

Für Perfektionisten wollen wir einen »intelligenten« Suchvorgang einbauen.

Was wollen wir als Abkürzung beziehungsweise als Fehler zulassen? Ein paar Beispiele für Nabburg:
Nabberg, Naburg, Naberg,
Nabb, Nab

Wir könnten sogar soweit gehen und festlegen, daß nur die ersten beiden Buchstaben stimmen müssen. Dann allerdings würden in unserem Fall alle neun Einträge der Liste ausgedruckt werden. Da aber im Postleitzahlenbuch nur eine begrenzte Anzahl von Städten aufgeführt ist, die mit »Na« anfangen, wäre diese Festlegung sinnvoll.

Wie muß nun der Suchvorgang ausschauen:

1. Der eingegebene Städtename NST$ wird mit allen Einträgen ST$(K) in der Liste verglichen.
2. Ist keine Übereinstimmung da, werden die beiden Namen NST$ und ST$(K) um ein Zeichen verkürzt und die Suche wiederholt.
3. Die Verkürzung wird fortgesetzt, bis die Namen nur noch aus zwei Buchstaben bestehen.
4. Ist immer noch keine Übereinstimmung vorhanden, wird eine Fehlermeldung ausgedruckt und eine neue Eingabe verlangt.
5. Sobald eine Übereinstimmung auftritt, bricht das Programm am Ende der betreffenden Suchschleife die weitere Suche ab und druckt das Ergebnis.

In Bild 18 ist der Vorgang grafisch dargestellt.

Es gab vorher nur eine Schleife, nämlich die für den Suchlauf durch die Liste der Einträge mit K als Zählvariable und Z als Obergrenze.

Jetzt brauchen wir eine zweite, übergeordnete Schleife, welche die Namenslänge nach jedem Durchlauf der K-Schleife reduziert. Diese Schleife mit der Variablen L beginnt ab der vollen Länge des eingegebenen Namens LEN(NST$) und zählt rückwärts bis zur Untergrenze von zwei Buchstaben:

```
 80 FOR L = LEN(NST$) TO 2 STEP -1
 85 FOR K=1 TO Z
 95 NEXT K
105 NEXT L
```

Die Prüfung auf Übereinstimmung in der alten Zeile 90 muß jetzt der steten Reduzierung der Namenslänge Rechnung tragen. Wir erreichen dies mit dem ebenso steten Abschneiden von Buchstaben von links her – natürlich mit dem LEFT$-Befehl.

```
 90 IF LEFT$(NST$,L) = LEFT$ (ST$(K),L)
    THEN PRINT PLZ$(K), ST$(K)
110 PRINT "STADT NICHT IN DER LISTE"
```

Vergleichen Sie bitte die Zeile 90 im oberen Abschnitt mit dieser hier. Sie werden den Unterschied sehen.

Jetzt fehlt nur noch der Abbruch, wenn eine Übereinstimmung gefunden worden ist. Er könnte in Zeile 90 nach dem Ausdrucken der Übereinstimmung durch einen Rücksprung auf die Namenseingabe in Zeile 75 erfolgen. Dann aber verlieren wir alle anderen noch möglichen Übereinstimmungen mit dem Rest der Liste.

NST$	HACKENHEIM	
ST$(2)	NACHTSHEIM	6 Treffer
ST$(3)	NACK	3 Treffer
ST$(4)	NACKENHEIM	9 Treffer

Bild 19. Buchstabenvergleich mehrerer Strings

NST$	GOLD	Treffer
ST$(6)	NAGOLD	
VST$:	---N AGOLD---	0
	---NA GOLD---	0
	---NAG OLD---	0
	--- NAGOLD---	0
	---N AGOLD---	0
	---NA GOLD---	4
	---NAG OLD---	0
	---NAGO LD---	0
	---NAGOL D---	0
	XST$	

Bild 20. Vergleich durch Verschiebung der Buchstaben

Logischerweise muß daher der Abbruch jeweils am Ende einer K-Schleife erfolgen.

Wir markieren die gefundene Übereinstimmung in Zeile 90 durch »Setzen einer Flagge« T:

```
90 IF LEFT$(NST$,L) = LEFT$ (ST$(K),L)
   THEN PRINT PLZ$(K), ST$(K) : T=1
```

Natürlich müssen wir diese Flagge T ganz am Anfang auf Null setzen:

```
70 T=0
```

Dann prüfen wir am Ende der K-Schleife, ob die Flagge gesetzt ist, wenn ja, dann Abbruch, wenn nein, dann nächste Zeichenreduzierung.

```
100 IF T=1 THEN 70
```

Das komplette Programm gibt Listing 21 wieder.

Wenn Sie jetzt einen Städtenamen korrekt eingeben, erhalten Sie auch nur eine Antwort. Geben Sie aber zum Beispiel den abgekürzten Namen »Nah« ein, druckt das Programm sowohl »Nahe« als auch »Nahrendorf«. Bei fehlerhafter Eingabe, nämlich »Nackenberg«, findet das Programm trotzdem »Nackenheim«. Wenn Sie nur NA eingeben, tritt das ein, was ich vorausgesagt habe, es werden alle neun Einträge, die mit »Na« anfangen, ausgegeben.

```
10 REM****** SORTIEREN 2 **********
12 :                                          <244>
20 Z=0                                        <021>
25 READ PLZ$,ST$                              <114>
30 IF PLZ$="@" THEN 45                        <143>
35 Z=Z+1                                      <103>
40 GOTO 25                                    <050>
45 DIM PLZ$(Z),ST$(Z)                         <020>
50 RESTORE                                    <100>
55 FOR I=1 TO Z                               <046>
60 READ PLZ$(I),ST$(I)                        <141>
65 NEXT I                                     <149>
67 :                                          <043>
70 T=0                                        <047>
75 INPUT "NAME DER STADT";NST$                <000>
80 FOR L=LEN(NST$) TO 2 STEP-1                <096>
83 :                                          <059>
85 FOR K=1 TO Z                               <092>
90 IF LEFT$(NST$,L) = LEFT$(ST$(K),L) THEN
   PRINT PLZ$(K),ST$(K):T=1                   <255>
95 NEXT K                                     <195>
98 :                                          <074>
100 IF T=1 THEN 70                            <057>
105 NEXT L                                    <213>
110 PRINT"STADT NICHT IN DER LISTE"           <028>
998 :                                         <212>
999 :                                         <213>
1000 DATA 8470,NABBURG,5441,NACHTSHEIM        <058>
1005 DATA 6509,NACK,6506,NACKENHEIM           <218>
1010 DATA 8590,NAGEL,7270,NAGOLD,2061,NAHE    <221>
1015 DATA 2121,NAHRENDORF,8674,NAILA          <213>
2000 DATA @,@                                 <184>
```

Listing 21. Das komplette Sortierprogramm



Das letzte Beispiel zeigt auch schon den Schönheitsfehler dieses Suchprogramms:

Die ersten beiden Buchstaben müssen stimmen. Ist nur einer der beiden falsch und alles andere richtig, wird keine Übereinstimmung gefunden. Wir werden darauf im nächsten Abschnitt näher eingehen. Fürs erste soll uns das jetzige Programm genügen, das Sie noch durch eigene Einträge ergänzen und weiter ausprobieren können.

6. Suchverfahren, Wörterraten

In diesem Abschnitt wollen wir uns zunächst noch einmal mit der im vorherigen Teil entwickelten Sortiermethode befassen, die im Suchprogramm noch nicht ganz perfekt war: Für eine Übereinstimmung mußten stets die ersten beiden Buchstaben richtig sein.

Dieser Nachteil müßte sich eigentlich irgendwie beheben lassen, wenn wir die Buchstaben einzeln miteinander vergleichen.

Bild 19 zeigt, wie das gemeint ist. In einem buchstabenweisen Vergleich werden die Übereinstimmungen am richtigen Platz gezählt. Der Städtename mit der höchsten Trefferzahl wird ausgedruckt. Oder besser noch, die Trefferzahl, bezogen auf die Länge des eingegebenen Namens NST$, wird in Prozent ausgerechnet und alle Datenpaare, deren Wert über einer bestimmten (wählbaren) Prozentzahl liegen, werden ausgegeben.

Der ganze erste Teil des Listings 21 aus Lektion 5, nämlich bis zur Zeile 65, und die DATA-Zeilen werden übernommen.

```
70 INPUT "NAME DER STADT";NST$
75 FOR K=1 TO Z
110 NEXT K
```

Auch die Eingabe und die K-Schleife für das Durchsuchen der Liste tauchen hier wieder auf, nur mit anderen Zeilennummern. Jetzt ist die K-Schleife die äußere Schleife. Pro Datenpaar der Liste werden in der inneren L-Schleife

Buchstabenvergleich von Strings

die einzelnen Buchstaben der beiden Namen NST$ und ST$ miteinander verglichen.

```
85 FOR L=1 TO LEN(NST$)
90 IF MID$(NST$,L,1) = MID$ (ST$(K),L,1) THEN X=X+1
95 NEXT L
```

Die innere Schleife hat soviel Durchläufe wie der eingegebene Namen Buchstaben hat (Zeile 85). Das ist durchaus nicht zwingend. Man könnte als obere Grenze auch die Länge des jeweiligen Namens in der Liste nehmen, oder aber eine fest vorgegebene Zahl. Ich bleibe vorerst bei der Methode der Zeile 85.

Den eigentlichen Vergleich machen wir in Zeile 90 mit dem Befehl MID$(NST$,L,1). Dabei zählt L die einzelnen Buchstaben. Es wird immer nur ein Buchstabe genommen. Wenn die Buchstaben gleich sind, wird ein Treffer X gezählt. Dieser Zähler muß am Anfang auf 0 gestellt werden.

```
80 X=0
```

Zur Sichtbarmachung der Vergleichsresultate füge ich noch einen Ausdruck ein:

```
96 PRINT X;ST$(K)
```

```
10 REM****** SUCHEN V.1 ***********
12 :                                        <244>
20 Z=0                                      <021>
25 READ PLZ$,ST$                            <114>
30 IF PLZ$="@" THEN 45                      <143>
35 Z=Z+1                                    <103>
40 GOTO 25                                  <050>
45 DIM PLZ$(Z),ST$(Z)                       <020>
50 RESTORE                                  <100>
55 FOR I=1 TO Z                             <046>
60 READ PLZ$(I),ST$(I)                      <141>
65 NEXT I                                   <149>
67 :                                        <043>
70 INPUT "NAME DER STADT";NST$              <251>
75 FOR K=1 TO Z                             <082>
80 X=0                                      <073>
85 FOR L=1 TO LEN(NST$)                     <050>
90 IF MID$(NST$,L,1) = MID$(ST$(K),L,1) TH
   EN X=X+1                                 <035>
95 NEXT L                                   <203>
96 PRINT X;ST$(K)                           <046>
98 :                                        <074>
100 PR=100/LEN(NST$)*X                      <170>
105 IF PR>=70 THEN PRINT ,,PLZ$(K)" "ST$(K
   );PR"%"                                  <016>
108 :                                       <084>
110 NEXT K                                  <210>
998 :                                       <212>
999 :                                       <213>
1000 DATA 8470,NABBURG,5441,NACHTSHEIM      <058>
1005 DATA 6509,NACK,6506,NACKENHEIM         <218>
1010 DATA 8590,NAGEL,7270,NAGOLD,2061,NAHE  <221>
1015 DATA 2121,NAHRENDORF,8674,NAILA        <213>
2000 DATA @,@                               <184>
```

Listing 22. Suchprogramm mit Entscheidungsschwelle

Wir drucken damit die Anzahl der Treffer und den jeweiligen Namen aus.

Starten Sie mal das Programm und geben als erstes den fehlerhaften Ortsnamen von Bild 19, nämlich Hackenheim ein. Sie werden das gleiche Ergebnis wie in dem Bild erhalten. Bei Eingabe der Abkürzung »Nah« haben »Nahe« und »Nahrendorf« die höchste Trefferzahl. Bei »Na« erscheint erwartungsgemäß kein Unterschied, da alle Namen in der Liste das Suchkriterium gleich gut erfüllen.

Jetzt wollen wir noch die Prozentschwelle einbauen. Die Prozente errechnen sich nach der Formel:

Treffer / Buchstabenzahl * 100

```
100 PR = (X/LEN(NST$))*100
105 IF PR >= 70 THEN PRINT,,PLZ$(K)" " ST$(K);PR"%"
```

Ich empfehle Ihnen, wie in Zeile 100 praktiziert, bei mathematischen Ausdrücken immer Klammern zu verwenden, um etwaige Unsicherheiten in der Reihenfolge der Berechnung auszuschließen.

Auch in Zeile 105, in der die Schwelle auf 70 Prozent gelegt ist, sind zwei grafische Anmerkungen zu machen. Zum ersten stehen direkt nach dem PRINT-Befehl zwei Kommata, die den Ausdruck auf die rechte Bildschirmhälfte schieben. Zum zweiten ist zwischen der Postleitzahl PLZ$ und dem Namen ST$ eine Leerstelle eingeschoben, da in Strings selbst Zahlen keine Leerstellen beinhalten.

Das komplette Programm ist als Listing 22 zusammengefaßt. Es bleibt Ihnen überlassen, die Zeile 96 zu entfernen, sobald Sie die Trefferzählung verstanden haben. Der Ausdruck dieser Zeile ist für das Ergebnis nicht wichtig.

Dieses Suchprogramm scheint recht komfortabel und brauchbar zu sein und hebt sich durch die einstellbare Entscheidungsschwelle von anderen Lösungen ab. Es hat aber leider immer noch eine kleine Schwäche. Um sie zu sehen, geben Sie als Ortsnamen das Fragment »Gold« ein. Sie erhalten keinen einzigen Treffer, obwohl bei dem Ortsnamen »Nagold« vier von sechs Buchstaben richtig sind. Nur, sie werden nicht erkannt, weil die eingegebenen Buchstaben, die am Anfang des Wortes stehen, nicht mit den richtigen Buchstaben verglichen werden.

Dieser Fehler führt uns zwangsläufig zu einer weiteren Suchmethode, bei der die zu vergleichenden Wörter gegeneinander verschoben werden.

In Bild 20 ist das Schema dieses Vergleichs dargestellt. Ich meine, die eindeutige Spitze in der Trefferquote spricht für die Methode. Wenn Sie das gleiche Spiel mit dem einzigen anderen Namen in der Liste machen, das mit »Gold« noch eine gewisse Ähnlichkeit hat, nämlich mit »Nagel«, dann sieht das Ergebnis ähnlich aus, aber mit deutlich weniger Treffern (nur 2).

Obwohl mit dieser Methode der Suchvorgang wesentlich verlängert wird, scheint das Ergebnis den Aufwand zu rechtfertigen.

Vergleich durch Wortverschiebung

Die Basis des Programms bilden wieder Teile des Listing 22, nämlich die Zeilen 20 bis 65 und die DATA-Listen der Zeilen 1000 bis 2000. Sie dienen dem Speichern der Datenpaare (Postleitzahl und Städtenamen) in ein Feld und der Bestimmung der Anzahl von Eintragungen Z in der Liste.

Zuerst geben wir wieder den Namen einer Stadt NST$ (oder seine Abkürzung) in Zeile 75 ein. Der Suchvorgang wird wie vorher von der Schleife mit der Variablen K gesteuert, wobei die Zahl Z die obere Grenze der Schleifenvariablen bildet.

```
75 INPUT "NAME DER STADT";NST$
80 FOR K=1 TO Z
180 NEXT K
```

Wenn wir auf Bild 20 schauen, sehen wir die notwendigen Schritte:

1. Die Anzahl der Schiebeschritte ist die Summe der Buchstabenzahl beider Wörter NST$ plus ST$(K) minus 1
   V = LEN(NST$) + LEN(ST$(K)) - 1
2. Damit der erste Schritt den ersten Buchstaben von ST$(K) mit dem letzten Buchstaben des eingegebenen Namens NST$ vergleicht, müssen vor ST$(K) Leerzeichen eingefügt werden und zwar um 1 weniger als die Wortlänge von NST$.
3. Für den letzten Schritt gilt das gleiche am Ende des Wortes.
4. Für jeden dieser Verschiebeschritte muß – wie im Programm vorher – ein buchstabenweiser Vergleich durchgeführt werden.

Zuerst wollen wir die Punkte 2 und 3 erfüllen. Wir müssen also Leerstellen vor und hinter das jeweilige Wort ST$(K) setzen. Die Anzahl der Leerstellen ist abhängig von der Länge des eingegebenen Namens NST$ und errechnet sich aus:

LEN(NST$)-1

Diesen Leerzeichen-String erzeugen wir mit einer kleinen Schleife:

```
85 A$ = " "
90 FOR A=1 TO LEN(NST$)-1
```




```
95 A$=A$+" "
100 NEXT A
105 VST$ = A$ + ST$(K) + A$
```

Da diese Worterweiterung für alle Namen ST$(K) in der Liste gemacht werden muß, deren Länge nicht immer gleich ist, wird der Leerzeichen-String A$ am Anfang auf Null gesetzt (Zeile 85). Der in den Zeilen 90 bis 100 erzeugte String A$ wird in Zeile 105 vor und hinter das Wort ST$(K) gesetzt und bildet so das neue erweiterte Wort VST$.

Wenn Sie diese Wortbildung überprüfen wollen, dann setzen Sie vorübergehend in Zeile 95 einen Punkt zwischen die Gänsefüße und geben einen zusätzlichen Druckbefehl ein, den wir aber später nicht mehr brauchen:

```
95 A$=A$+"."
106 PRINT VST$
```

Wenn alles läuft, löschen Sie bitte Zeile 106.

Den Kern bildet wieder das buchstabenweise Vergleichen, so wie wir es im vorigen Programm schon in den Zeilen 85 bis 95 gemacht haben. Wir verwenden wieder die gleiche L-Schleife. Nur dürfen wir nicht das ganze Wort ST$(K) zum Vergleich nehmen, sondern nur den sich stetig ändernden Teil, der in Bild 19 direkt unter NST$ (Gold) steht. Ich nenne ihn XST$.

```
140 FOR L=1 TO LEN((NST$)
145 IF MID$(NST$,L,1) = MID$(XST$,L,1) THEN X=X+1
150 NEXT L
```

Das neue Wort wird, wie aus Bild 20 ersichtlich, während der Verschiebung ständig neu gebildet. Deshalb wollen wir zunächst vor die L-Schleife, welche die Buchstaben vergleicht, die Verschiebeschleife mit der Variablen M setzen.

```
120 FOR M=1 TO V
175 NEXT M
```

Die obere Grenze der Variablen M, in Zeile 120 mit V angegeben, wird wie folgt gebildet:

```
115 V=LEN(NST$)+LEN(ST$(K))-1
```

Jetzt sind wir auch in der Lage, das Teilwort XST$ innerhalb der M-Schleife zu bilden, und zwar so:

```
135 XST$ = MID$(VST$,M,LEN(NST$))
```

Viele Treffer – wenig Aufwand

Für jeden Wert von M wird mit dem MID$-Befehl aus dem erweiterten Wort VST$ ab der M-ten Stelle ein Teil mit der Länge des eingegebenen Namens NST$ herausgeschnitten. Falls Sie unsicher sind, schauen Sie sich bitte noch einmal Bild 20 an. Da sieht man das sehr deutlich.

So, jetzt bleibt uns nur noch die Aufgabe, aus den gezählten Treffern der Zeile 145 eine Aussage zu machen. Auch

das ist identisch mit dem letzten Programm, es haben sich nur die Zeilennummern geändert.

```
130 X=0
160 PRINT X;XST$" "NST$
165 PR=(100/LEN(NST$))*X
170IF PR>=70 THEN PRINT ,,PLZ$(K)" " ST$(K);PR"%"
```

Auch hier habe ich die Schwelle auf 70 Prozent gelegt. Sie können durch Variieren dieser Schwelle ein Optimum herausfinden.

Die Zeile 160 dient wieder nur zum Verstehen des Ablaufs, sie kann später gelöscht werden.

Jetzt ist das Programm fertig. Es ist in richtiger Reihenfolge komplett in Listing 23 wiedergegeben.

Wenn Sie nach RUN das Fragment »Nah« eingeben, erhalten Sie zwei gleichwertige Antworten, nämlich »Nahe« und »Nahrendorf«. Bei »Gold« kommt »Nagold« als eindeutiges Resultat heraus. Bei »Mackesheim« müssen Sie selbst zwischen 70 Prozent für »Nachtsheim« und 80 Prozent für »Nackenheim« wählen.

Beim letzten Beispiel wird allerdings der Nachteil dieser Suchmethode deutlich: Sie benötigt recht viel Zeit.

Diese Methode des Vergleichens mit schrittweiser Verschiebung hat in der modernen Wissenschaft und Technik eine große Bedeutung. Sie wird »Korrelation« genannt und dient vor allem dem Herausfiltern von extrem schwachen Signalen aus einer störenden Umgebung.

Sie ist aber auch die grundlegende Methode, mit der in Textprogrammen und ganz besonders in Adventure-Programmen eingegebene Sätze und Anweisungen grammatisch zerlegt und analysiert werden, um dann irgend eine intelligente Antwort geben zu können. Derartige Programme haben den englischen Namen »Parser«.

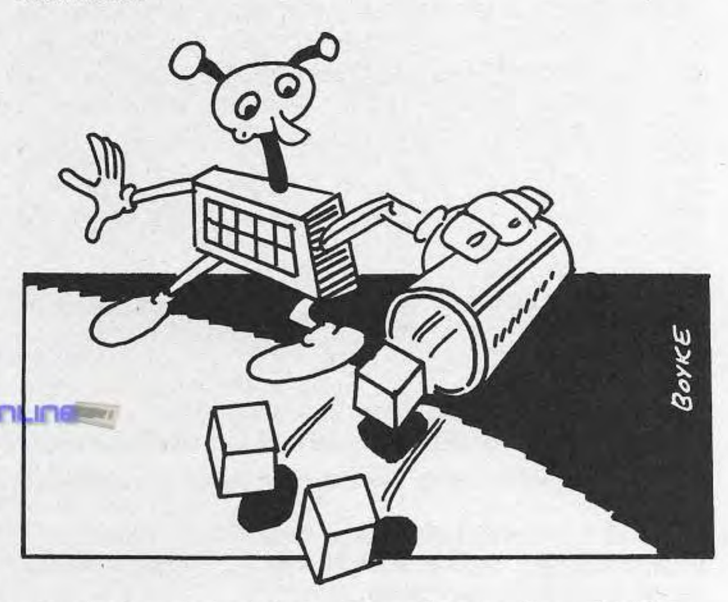
In den 64'er-Sonderheften 2/85 und 4/86 wurden zwei detaillierte Kurse von Michael Nickles über die Programmierung von Adventure-Programmen veröffentlicht. Darin werden viele interessante String-Anwendungen für Befehlseingabe, Anlegen von Listen in relativen Dateien, Codieren von Worttypen und so weiter gezeigt.

```
10 REM****** SUCHEN.KOMPL *******
15 :                                          <247>
20 Z=0                                        <021>
25 READ PLZ$,ST$                              <114>
30 IF PLZ$="@" THEN 45                        <143>
35 Z=Z+1                                      <103>
40 GOTO 25                                    <050>
45 DIM PLZ$(Z),ST$(Z)                         <020>
50 RESTORE                                    <100>
55 FOR I=1 TO Z                               <046>
60 READ PLZ$(I),ST$(I)                        <141>
65 NEXT I                                     <149>
70 :                                          <046>
75 INPUT "NAME DER STADT";NST$                <000>
80 FOR K=1 TO Z                               <087>
82 :                                          <058>
85 A$=""                                      <114>
90 FOR A=1 TO LEN(NST$)-1                     <097>
95 A$=A$+" "                                  <156>
100 NEXT A                                    <120>
105 VST$=A$+ST$(K)+A$                         <066>
110 :                                         <086>
115 V=LEN(NST$)+LEN(ST$(K))-1                 <239>
120 FOR M=1 TO V                              <141>
130 X=0                                       <123>
135 XST$=MID$(VST$,M,LEN(NST$))               <220>
138 :                                         <114>
140 FOR L=1 TO LEN(NST$)                      <105>
145 IF MID$(NST$,L,1) = MID$(XST$,L,1) THE
    N X=X+1                                   <199>
150 NEXT L                                    <002>
152 :                                         <128>
160 PRINT X;XST$" "NST$                       <174>
165 PR=(100/LEN(NST$))*X                      <075>
170 IF PR>=70 THEN PRINT ,,PLZ$(K)" "ST$(K
    );PR"%"                                   <081>
172 :                                         <148>
175 NEXT M                                    <035>
177 :                                         <153>
180 NEXT K                                    <024>
190 :                                         <166>
195 :                                         <171>
1000 DATA 8470,NABBURG,5441,NACHTSHEIM        <058>
1005 DATA 6509,NACK,6506,NACKENHEIM           <218>
1010 DATA 8590,NAGEL,7270,NAGOLD,2061,NAHE    <221>
1015 DATA 2121,NAHRENDORF,8674,NAILA          <213>
2000 DATA @,@                                 <184>
```

Listing 23. Das komplette Suchprogramm

Satzzerlegungen ohne Probleme

Herr Nickles ist nicht nur der Autor des Adventure-Programms Gordon Saga, sondern auch der Autor eines ausgezeichneten Parsers. Da die Kurse sehr ausführlich und leicht verständlich sind, kann ich mich hier zu diesem Thema kurz fassen und Sie, liebe Leser, auf die beiden Kurse verweisen.



Ein kleines Beispiel aus diesen Kursen aber will ich hier doch zitieren.

Bei einem Adventure werden bekanntlich vom Spieler laufend Befehle eingegeben, die vom Parser dann zerlegt und analysiert, das heißt mit den im Spiel vorhandenen Wörtern verglichen werden.

Diese Befehlseingabe besteht im Gegensatz zu unseren bisherigen Beispielen aus mehreren Wörtern:

```
BE$="OTTO BITTE NIMM ES"
```

Wir wissen natürlich, daß das Verb dieses Satzes das Wort »nimm« ist. Der Parser muß das erst feststellen. Dazu holt er aus einer im Computer (oder auf der Diskette) gespeicherten Verben-Liste der Reihe nach die einzelnen Verben und vergleicht sie mit dem Satz BE$. Ich konzentriere mich hier auf den Vergleich. Die Schleifen zum Durchlaufen der Listen sind gleich aufgebaut wie in unseren vorigen Programmen.

Auch der direkte Vergleich ist ähnlich wie vorher. Nehmen wir an, aus der Liste sei das Verb VG$= "NIMM" an der Reihe. Bild 21 zeigt den Suchvorgang.

Das Programm dazu lautet:

```
10 X=0
20 BE$="OTTO BITTE NIMM ES"
30 INPUT "WORT EINGEBEN";VG$
40 FOR I=1 TO LEN(BE$)
50 IF VG$=MID$(BE$,I,LEN(VG$)) THEN X=1
60 NEXT I
70 IF X=0 THEN PRINT "FEHLANZEIGE"
80 IF X=1 THEN PRINT "WORT GEFUNDEN"
```

Eigentlich ist das nichts Neues. Es ist halt nur ein Wortvergleich, nicht ein absoluter Buchstabenvergleich. Das kann man natürlich ausnutzen, um die Suchschleife zu beschleunigen, indem die Obergrenze für die Schleifenvariable I verkleinert wird. Wenn Sie sich Bild 21 anschauen, dann werden Sie erkennen, daß das zu suchende Wort spätestens in den letzten, seiner Länge entsprechenden Zeichen des Satzes auftreten muß. Zeile 40 kann demnach so geändert werden:

```
40 FOR I=1 TO LEN(BE$)-LEN(VG$)+1
```

Diese Suchmethode kann natürlich auch Wortfragmente und Abkürzungen verarbeiten, aber auch Fehler. Im oben erwähnten Adventurekurs wird darauf näher eingegangen.

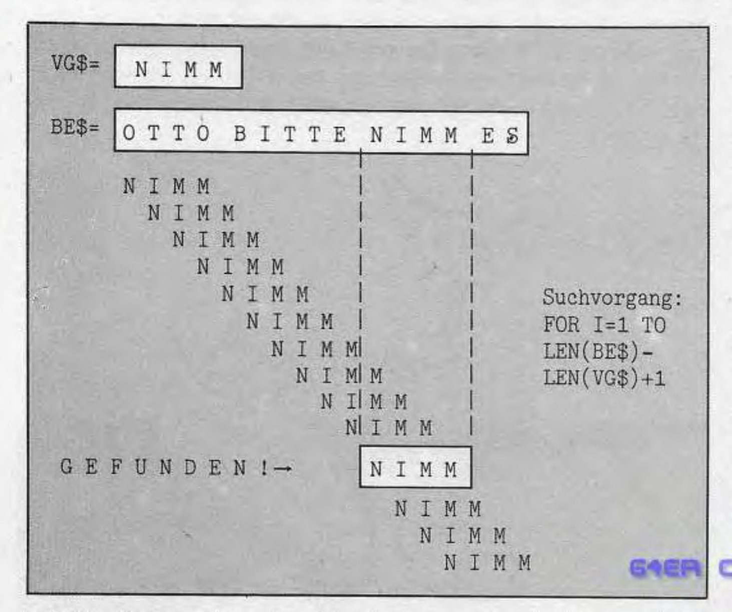


Bild 21. Beispiel für einen Suchvorgang durch Wortvergleich

Ich will zum Schluß mit einer anderen String-Anwendung Ihren Spieltrieb fördern. Zum Thema »Chiffrieren« wollen wir ein kleines Spiel programmieren.

Unter Verschlüsseln oder Chiffrieren versteht man das Verändern eines Textes derart, daß ein Fremder ihn nicht versteht, wohl aber ein Freund, der den »Schlüssel« dazu hat.

Nun, ganz so geheimnisvoll will ich hier nicht vorgehen. Mein Programmplan sieht so aus:

1. Spieler Nummer 1 gibt ein Wort in den Computer ein.
2. Die einzelnen Buchstaben dieses Wortes werden von einem Zufallsgenerator gesteuert, gegeneinander vertauscht und das so »chiffrierte« Wort wird dem Spieler Nummer 2 gezeigt.
3. Spieler Nummer 2 soll das ursprüngliche Wort erraten. Er hat dazu beliebig viele Versuche, kann aber auch eine Hilfestellung verlangen.

Das eingegebene Wort A$ wird zuerst auf die Variable B$ übertragen. A$ wird ja durch die Verschlüsselung verändert. Deshalb speichern wir das Original dieses Wortes in B$ für den Rate- oder Vergleichsvorgang. Die Länge des Wortes ordnen wir der leichteren Lesbarkeit den Variablen LA und LB zu. H ist der Zähler für die Anzahl der Hilfestellungen; er wird am Anfang auf Null gestellt.

```
65 Z=INT(RND(0)*LA)+1
70 C$=C$+MID$(A$,Z,1)
```

4. Zur Hilfestellung werden ihm der erste, beim zweiten Mal die ersten beiden Buchstaben und so weiter gezeigt. Dadurch ist die Zahl der Hilfen natürlich begrenzt.

Um das Programm einfach zu halten, verzichte ich auf Ergebnislisten und andere Feinheiten, die Sie ja selbst nach Gutdünken hinzufügen können.

Zuerst kommt die Eingabe und die Chiffrierung an die Reihe:

```
35 INPUT "WORT EINGEBEN";A$
40 B$=A$
45 LA=LEN(A$)
50 LB=LEN(B$)
55 H=0
```

Der Zufallsgenerator, der die gewünschte Chiffrierung steuert, steht in Zeile 65. Er erzeugt eine ganzzahlige Zufallszahl, die zwischen 1 und der Wortlänge des eingegebenen Wortes A$ liegt. In Zeile 70 erfolgt die eigentliche Chiffrierung: Es wird ein neuer String C$ aufgebaut. Er wird aus seinem alten Wert und jeweils einem neuen Buchstaben; gebildet. Dieser String wird per MID$-Befehl ab der vom Zufall bestimmten Stelle aus dem Wort A$ herausgeschnitten.

Damit ein bereits verwendeter Buchstabe des Strings A$ nicht ein zweites Mal benützt werden kann, muß nach jedem Schritt dieser Buchstabe aus dem Wort A$ entfernt werden.

```
75 A$=LEFT$(A$,Z-1)+RIGHT$(A$,LA-Z)
80 LA=LA-1
85 IF LA <> 0 THEN 65
90 PRINT C$
```

Zeile 75 verändert das Wort A$. Es besteht jetzt aus dem Teil links vom herausgenommenen Buchstaben (Z-1) und aus dem Teil rechts dieses Buchstabens (LA-Z). Bild 22 zeigt den Zusammenhang.

Wenn Sie in der Zeile 70 mit Doppelpunkt getrennt noch den Befehl

```
PRINT C$
```

anhängen und das bisherige Programm starten, sehen Sie die Wirkung auf dem Bildschirm.

Der nächste Programmteil ist einfach. Er beinhaltet nur die Abfrage, ob geraten werden soll oder ob Hilfe gebraucht wird. Der Antwort entsprechend wird auf bestimmte Programmteile verzweigt.

```
105 PRINT "RATEN ODER HILFE ? (R/H)
110 GET X$: IF X$="" THEN 110
120 IF X$="R" THEN 170
130 IF X$<>"H" THEN 105
```

Z=2	WORT	LA=4	
Z=3	WRT	LA=3	O
Z=2	WR	LA=2	OT
Z=1	W	LA=1	OTR
	—	LA=0	OTRW

Bild 22. Die Chiffrierschritte im Zusammenhang

Nach der Aufforderung, <R> oder <H> zu drücken, wartet Zeile 110 auf einen Tastendruck. Zeile 120 springt beim Drücken von <R> auf den Rateteil. Zeile 130 springt

Wörterraten leichtgemacht

bei allen anderen Eingaben außer <H> auf die Aufforderung in Zeile 105 zurück und verhindert dadurch falsche Eingaben. Ab Zeile 140 beginnt die Hilfestellung.

```
140 H=H+1
145 IF H=LB-1 THEN 195
150 PRINT C$,,LEFT$(B$,H)
160 GOTO 105
195 PRINT "DAS WORT WAR " B$
```

Da Hilfe in Anspruch genommen wird, wird zuerst der Zähler H um 1 erhöht. Wenn die Anzahl der Hilfen das Wortende, hier gegeben durch LB-1 (LA haben wir ja oben dauernd reduziert) erreicht haben, wird das Spiel in Zeile 195 abgebrochen.

Zeile 150 gibt die Hilfestellung. Zuerst wird das chiffrierte Wort C$ noch einmal ausgedruckt. Danach werden die Buchstaben entsprechend dem Wert von H ausgedruckt. Der Sprung auf Zeile 105 wiederholt die Frage nach R oder H. Damit ist der Hauptteil des Programms abgeschlossen.

Jetzt fehlt nur noch der Ratevorgang.

```
170 INPUT "WAS RATEN SIE ";R$
175 IF R$<>B$ THEN PRINT "NEIN":GOTO 105
180 IF R$=B$ THEN PRINT "RICHTIG"
```

In Zeile 170 wird gefragt, welches Wort geraten wird, in Zeile 175 folgt der Wortvergleich. Diese Befehle sind aber so einfach, daß ich guten Gewissens die Kommentierung weglassen kann.

Das komplette Programm ist in Listing 24 zusammengefaßt.

7. Menüs, Tabellen, Listen

Zum Abschluß unseres String-Kurses wollen wir ein Programm entwickeln, das sich natürlich ausgiebig auf String-Verarbeitungen stützt und außerdem Einblick gewährt, wie die in vielen kommerziellen Computerspielen eingebauten »Bestenlisten« funktionieren.

Wie immer werde ich das Programm in einzelnen Schritten entwickeln. Da jeder Teil für sich lauffähig ist, können Sie wie im Text die einzelnen Programmzeilen eintippen und ausprobieren. Ich rate Ihnen nur, die von mir gewählten Zeilennummern zu verwenden, da sie im Zusammenhang mit dem späteren endgültigen Programm stehen.

Die Bestenlisten – in englischsprachigen Spielen »High Score« genannt – enthalten die besten Ergebnisse, die in einem Spiel erzielt worden sind. Dabei gibt es drei prinzipielle Unterschiede:

```
10 REM***** WOERTERRATEN ******
15 :                                        <247>
20 REM ---- CHIFFRIEREN-RATEN ------        <099>
25 :                                        <001>
30 PRINT CHR$(147)                          <059>
35 INPUT "WORT EINGEBEN";A$                 <130>
40 B$=A$                                    <174>
45 LA=LEN(A$)                               <043>
50 LB=LEN(B$)                               <185>
55 H=0                                      <240>
60 :                                        <036>
65 Z=INT(RND(0)*LA)+1                       <178>
70 C$=C$+MID$(A$,Z,1)                       <126>
75 A$=LEFT$(A$,Z-1)+RIGHT$(A$,LA-Z)         <046>
80 LA=LA-1                                  <149>
85 IF LA<>0 THEN 65                         <218>
90 PRINT C$                                 <028>
95 PRINT                                    <197>
100 :                                       <076>
105 PRINT "RATEN ODER HILFE ? (R/H)"        <144>
110 GET X$:IF X$="" THEN 110                <115>
115 :                                       <091>
120 IF X$="R" THEN 170                      <162>
125 :                                       <101>
130 IF X$<>"H" THEN 105                     <003>
135 :                                       <111>
140 H=H+1                                   <103>
145 IF H=LB-1 THEN 195                      <122>
150 PRINT C$,,LEFT$(B$,H)                   <146>
155 PRINT                                   <001>
160 GOTO 105                                <248>
165 :                                       <141>
170 INPUT"WAS RATEN SIE ";R$                <010>
175 IF R$<>B$ THEN PRINT"NEIN":GOTO 105     <145>
180 IF R$=B$ THEN PRINT "RICHTIG"           <057>
190 :                                       <166>
195 PRINT "DAS WORT WAR " B$                <078>
```

Listing 24. Spiel: Wörterraten mit dem C64

- die ewige Bestenliste
- die persönliche Bestenliste
- die Tages-Bestenliste.

1. Die ewige Bestenliste beruht darauf, daß der jeweils letzte Stand am Ende eines Spieldurchganges auf Diskette oder Kassette gespeichert wird. Zu Beginn einer neuen Spielrunde wird er geladen und steht für die Eintragung neuer Rekorde zur Verfügung.
2. Die persönliche Bestenliste ist eigentlich eine »ewige« Liste. Das heißt, sie wird auch gespeichert. Sie ist aber im Unterschied zu der ewigen Bestenliste so sortiert, daß jeder Name nur einmal enthalten ist, natürlich mit dem von diesem Spieler erzielten höchsten Ergebnis.
3. Die Tages-Bestenliste ist identisch mit der ewigen Bestenliste, nur wird sie nicht gespeichert, sondern enthält lediglich das Ergebnis einer Spielrunde. Sie sehen, es dreht sich wieder einmal (fast) alles ums Sortieren.

Ich zeige Ihnen zuerst ein Beispiel der Bestenliste:

EWIGE BESTENLISTE		
1.	FRANZ	678
2.	FRANZ	567
3.	FRANZ	503
4.	WILHELM	445
5.	MARIA	440
6.	WILHELM	390
7.	FRANZ	377
8.	BETTINA	50
9.	MARIA	33
10.	GABY	9

PERSÖNLICHE BESTENLISTE		
1.	FRANZ	678
2.	WILHELM	445
3.	MARIA	440
4.	BETTINA	50
5.	GABY	9
6.		

Franz ist also eindeutig der Beste, vielleicht weil er häufig spielt. In der ewigen Liste hält er die drei ersten Plätze. Auch Maria hat gute Ergebnisse und steht zweimal in der ewigen Liste.

In der persönlichen Liste dagegen ist jeder der fünf Spieler nur einmal vertreten, natürlich mit seinem besten Ergebnis.

Damit liegen die Sortiervorschriften eigentlich schon fest.

Ewige Bestenliste: Ein neues Ergebnis wird mit der ersten Eintragung in der Liste verglichen.

Ist es größer, kommt es an dessen Position, alle folgenden Werte rücken um einen Platz weiter nach hinten, und der Vorgang wiederholt sich mit dem nächsten Wert in der Liste.

Ist es gleich oder kleiner, dann wird sofort der nächste Listenwert zum Vergleich hergenommen.

Der Name des Spielers, der dieses Ergebnis erreicht hat, wird an dieselbe Position in der Liste gebracht, unabhängig davon, wie oft er schon in der Liste steht. Damit kann ein Spieler die komplette Liste belegen.

Persönliche Bestenliste: Ein neues Ergebnis führt zuerst zu einer Prüfung, ob der Name des Spielers schon in der Liste steht. Ist der Name schon enthalten, wird das neue Ergebnis mit dem alten Wert desselben Spielers verglichen. Der kleinere Wert wird gelöscht, der größere wird wie oben neu einsortiert.

Ist der Name aber nicht enthalten, wird der Wert wie oben der Größe nach einsortiert.

Aus diesen Vorschriften, aber auch beim Betrachten der beiden Listen wird deutlich, daß wir uns für das Problem der Zuordnung von Namen, Platznummern und Ergebnissen etwas einfallen lassen müssen. Allein der Name »Franz« tritt mit vier verschiedenen Ergebnissen an vier Positionen auf. Das klingt schlimmer, als es ist.

Zuerst werde ich nur mit zehn Werten arbeiten, also können wir uns den DIM-Befehl sparen.

Das Sortierverfahren

Ein Sortierverfahren haben wir schon in Teil 5 unseres Kurses entwickelt. Dieses Grundprinzip, das ich »Bäumchen wechsle Dich« genannt habe, wenden wir auch hier wieder an. Einigen Routinen werden daher in abgewandelter Form benutzt.

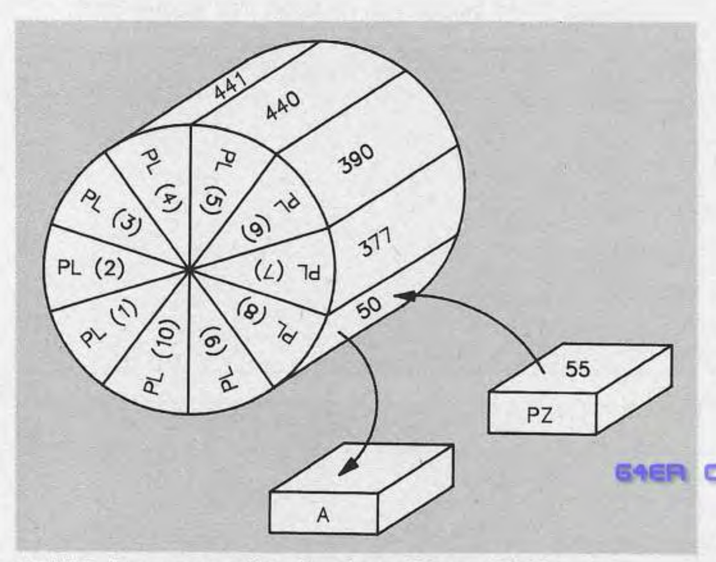


Bild 23. Das verwendete Sortierverfahren bildlich als Trommel veranschaulicht

Bei der Liste soll es sich um ein Feld handeln. Sie beinhaltet die Namen PL(K) – »Punkte-Liste«. Die Liste darf maximal zehn Werte PL(1) bis PL(10) enthalten.

Mit diesen Werten soll eine neue Punktzahl, die am Ende eines Spiels als Resultat PZ herausgekommen ist, verglichen werden. Anstelle eines Spieles simulieren wir das Ergebnis per INPUT-Befehl:

```
120 INPUT "RESULTAT=";PZ
```

Bekanntlich wird der Text in Gänsefüßchen hinter dem INPUT auf dem Bildschirm ausgedruckt.

Jetzt bilden wir eine Schleife mit der Variablen K von 1 bis 10, in der für jeden einzelnen Wert von K geprüft wird, ob das Resultat PZ größer als der im Feld stehende Wert PL(K) ist.

```
210 FOR K=1 TO 10
220 IF PZ >PL(K) THEN ....
230 NEXT K
250 GOTO 120
```

Ist PZ kleiner oder gleich, fährt die Schleife in Zeile 230 fort. Am Ende springt das Programm für eine neue Eingabe auf Zeile 120 zurück.

Ist PZ aber größer als der Wert PL(K) in der Liste (Zeile 220), dann muß etwas passieren. Was passiert, verdeutlicht Bild 23.

Dort ist das Feld PL(K) als Trommel dargestellt. Sie hat zehn Fächer, die PL(1) bis PL(10) heißen und in denen die Ergebnisse stehen. Vor der Trommel steht eine Schachtel mit dem Namen PZ, in der das letzte Spielergebnis liegt. Unter der Trommel gibt es noch eine zweite Schachtel mit Namen A, die vorerst leer ist.

Die Trommel wird durch die K-Schleife immer um ein Fach weitergedreht, und der Fachinhalt wird mit dem Inhalt der Schachtel PZ verglichen. Sobald der Inhalt der Schachtel – in unserem Beispiel die Zahl 55 – größer ist als der Inhalt des Trommelfaches, fällt dieser zuerst in die leere Schachtel A, dann wird der Inhalt der Schachtel PZ in das geleerte Trommelfach geschoben, und zuletzt gelangt der ehemalige Trommelinhalt in die Schachtel PZ. Dann dreht sich die Trommel weiter.

In Bild 23 ist dies zum erstenmal bei K=8 der Fall, da PZ=55 größer als PL(8)=50 ist. Am Ende der »Bäumchen wechsle Dich«-Prozedur steht in PL(8) die Zahl 55 und in PZ das »alte« PL(8), nämlich 50.

Dieses Wechselspiel wiederholt sich natürlich bei K=9 und K=10. Da sich danach die Trommel nicht mehr weiterdreht, geht der alte Wert von PL(10)=9 verloren.

Da diese Positionstauscherei immer wieder vorkommt, habe ich sie als kleines Unterprogramm ab Zeile 1100 geschrieben, auf das wir in der noch nicht vollständigen Zeile 220 springen.

```
120 INPUT "RESULTAT=";PZ
210 FOR K=1 TO 10
220 IF PZ >PL(K) THEN GOSUB 1100
230 NEXT K
250 GOTO 120
1100 A=PL(K)
1110   PL(K)=PZ
1120          PZ=A
1160 RETURN
```

Übrigens: Eine INPUT-Schleife mit direktem GOTO-Rücksprung ist gefährlich. Beim C 64 läßt sich das Programm nur mit <RUN/STOP-RESTORE> abbrechen. Daher füge ich die Abfragezeile 240 ein:

Die ewige Bestenliste

```
240 GET A$:IF A$="" THEN 240
```

Schließlich wollen wir uns das Resultat dieser Sortierung auch anschauen. Wir drucken nach jedem K-Schritt sowohl das K als auch den jeweiligen Trommelinhalt aus (Zeile 225):

```
225 PRINT K;PL(K)
```

Mit diesem Programmteil können Sie das Auffüllen der Liste schrittweise verfolgen.

```
100 REM******BASISPROGRAMM**********
102 :                                   <078>
105 REM----- EWIGE BESTENLISTE --------  <122>
107 :                                   <083>
110 INPUT "NAME";NA$                    <179>
120 INPUT"RESULTAT =";PZ                <236>
130 :                                   <106>
199 :                                   <175>
210 FOR K=1 TO 10                       <116>
220 IF PZ>PL(K) THEN GOSUB 1100         <210>
225 PRINT K;PL(K);NA$(K)                <240>
230 NEXT K                              <074>
240 GET A$:IF A$="" THEN 240            <146>
250 GOTO 110                            <194>
299 :                                   <021>
1100 A=PL(K)                            <084>
1110 PL(K)=PZ                           <147>
1120 PZ=A                               <206>
1125 :                                  <085>
1130 B$=NA$(K)                          <084>
1140 NA$(K)=NA$                         <224>
1150 NA$=B$                             <003>
1160 RETURN                             <202>
```


Listing 25. So sieht das Basisprogramm aus, das bei jeder Bestenliste verwendet wird



Dabei fallen folgende Punkte auf:
- Nach elf Eingaben geht der zehnte Wert verloren.
- Selbst wenn PZ einen Wert enthält, der bereits in der Liste vorkommt, wird er trotzdem mit aufgenommen. Die neue Eintragung steht hinter dem identischen, schon vorhandenen Wert.
- Wird ein neuer Wert irgendwo vor zwei gleichen Werten eingeschoben, vertauschen die beiden Zwillinge ihren Platz. Das kommt von der Platztauscherei dieses Sortierverfahrens. Um dies zu vermeiden, müßte man eine kompliziertere und umfangreiche Methode verwenden, deshalb verzichte ich darauf. Es stört ja auch kaum.

Die Sortiervorschrift erzeugt ja bereits die ewige Liste, aber nur für Zahlenwerte. Es fehlen dabei die Namen.

Nun, der Vorgang ist genau derselbe wie vorher, nur halt mit Strings statt mit Zahlen. Zunächst müssen wir den Namen (NA$) des Spielers eingeben:

```
110 INPUT "NAME";NA$
```

Was mit dem Namen passiert, zeigt Bild 24.

Sie zeigt für die Namen eine zweite Trommel, die aber mit der ersten Trommel fest verbunden ist und dadurch auf die gleiche Art und Weise gedreht wird. Immer dann, wenn eine Zahl PL(K) ausgetauscht wird, wird auch der dazugehörige Name NA$(K) ausgetauscht, und zwar mit derselben Methode. Auch vor dieser Trommel steht eine Schachtel NA$, die den Namen des Spielers enthält und eine zweite leere Schachtel B$, die zum Platztauschen verwendet wird.

Folglich müssen wir nur die zweite Trommel programmieren. Sie folgt auf die erste Trommel hinter Zeile 1120:

```
1130 B$=NA$(K)
1140      NA$(K)=NA$
1150             NA$=B$
```

Der RETURN-Befehl ist überflüssig. Er steht ja noch in Zeile 1160. Damit die durch diese Erweiterung des Unterprogramms erzeugte Namensliste NL$(k) auch sichtbar wird, ist der PRINT-Ausdruck in Zeile zu 225 zu erweitern:

```
225 PRINT K;PL(K);NA$(K)
```

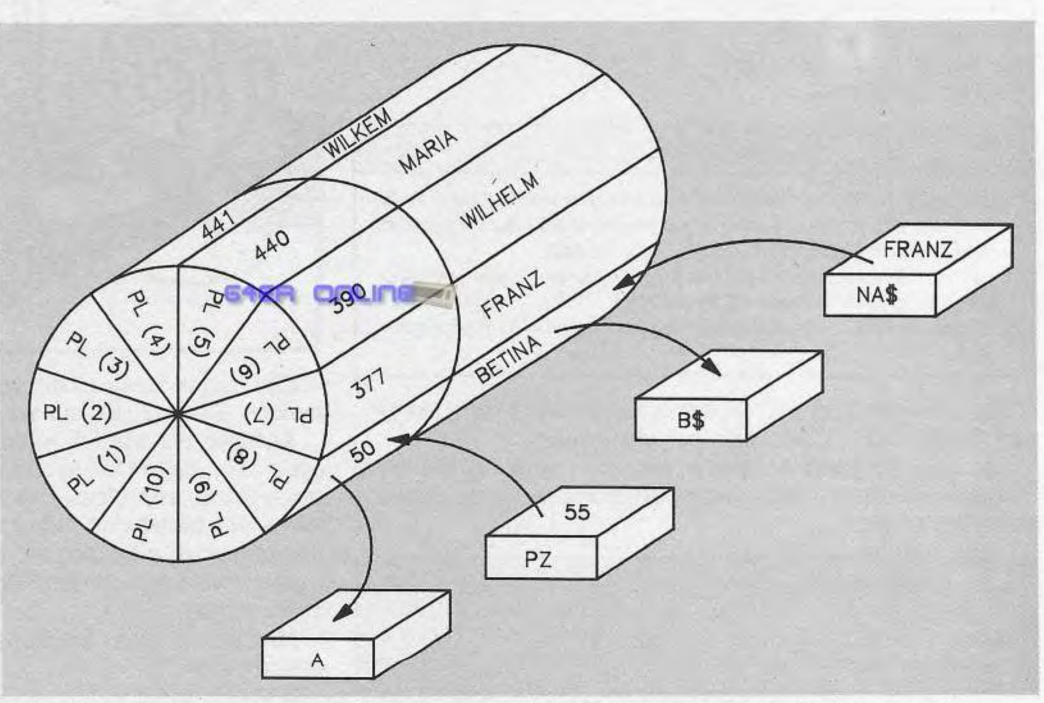


Bild 24. Um beim Sortieren die Namen zu berücksichtigen, ist eine weitere Trommel erforderlich. Sie ist mit der ersten Trommel fest verbunden

Als letztes müssen wir noch den Rücksprung der Zeile 250 umschreiben, denn wir beginnen das Programm jetzt mit der Namenseingabe in Zeile 110.

```
250 GOTO 110
```

Dieser Teil bildet die Basis für weitere Listen (Listing 25).

Die persönliche Bestenliste

So können wir das Programm als »ewige Bestenliste« (mit speichern) oder als »Tagesbestenliste« (ohne speichern) verwenden.

Ganz am Anfang habe ich definiert, was eine »persönliche Bestenliste« ist.

Sie wird zum größten Teil genauso aufgebaut wie die ewige Bestenliste, aber jeder Spielernamen soll nur einmal eingetragen sein.

Wir beginnen das Programm wie vorher. Dann prüfen wir für jede Eintragung in der Namensliste NA$ in einer Schleife mit zehn Schritten, ob der neue Spielernamen NA$ schon vorkommt.

```
150 FOR K=1 TO 10
160 IF NA$=NA$(K) THEN ....
180 NEXT K
```

Wenn der Name nicht übereinstimmt, läuft die Schleife weiter. Was passiert aber, wenn in Zeile 160 festgestellt wird, daß der Name schon in der Liste steht? Nun, dann kommt es als nächstes darauf an, ob das neue Spielergebnis PZ größer oder kleiner ist als der alte Wert PL(K).

Ist er kleiner oder gleich, dann kann man ihn »vergessen«, indem man ihn auf 0 setzt. So stört er nicht weiter.

```
160 IF NA$=NA$(K) AND PZ <=PL(K) THEN PZ=0
```

Ist er aber größer, dann muß er an die Stelle des alten Wertes gesetzt werden. Das geht am elegantesten dadurch, daß der alte Wert PL(K) samt seinem Spielernamen gelöscht und dann mit der Vergleichsschleife ab Zeile 210 am richtigen Platz eingesetzt wird.

```
170 IF NA$=NA$(K) AND PZ >PL(K) THEN PL(K)=
     0:NA$(K)=""
```

```
210 FOR K=1 TO 10
220 IF PZ >PL(K) THEN GOSUB 800
```

und so weiter.

Sie sehen, die beiden Listentypen unterscheiden sich nur durch die Zeilen 150 bis 180.

Das Programm für die persönliche Bestenliste ist in Listing 26 komplett dargestellt.

In einem Spiel angewendet, könnten wir beide Programmteile hintereinanderhängen.

Die Sache hat aber zwei Haken:

- Im Lauf der Schleifen und durch das Platztauschen im Unterprogramm verändert sich das ursprüngliche Ergebnis PZ, bis es zusammen mit dem Namen des Spielers verschwunden ist. Sie stehen also für die zweite Liste nicht mehr zur Verfügung.
- Wenn wir die zweite Liste ausrechnen und auf dem Bildschirm ausdrucken, geht die erste Liste verloren, da sie ja dieselben Variablennamen hat.

Zur Lösung des ersten Problems müssen wir das Ergebnis und den Spielernamen ganz am Anfang unter anderen Variablennamen speichern.

Ich wähle SN$ für den Spielernamen und RE für das Resultat. In Zeile 140 werden die beiden Werte den »alten« Variablennamen zugeordnet.

```
110 INPUT "NAME";SN$
120 INPUT "RESULTAT=";RE
140 NL$=SN$:PZ=RE
```

Zur Erinnerung

1. Ein eindimensionales Feld (englisch: array) ist eine Liste von Variablen desselben Namens, die mit nur einem »Index« durchnumeriert sind - zum Beispiel neun Variable A(0) bis A(8).
2. Ein zweidimensionales Feld hat für jede Variable zwei »Indizes«, zum Beispiel 18 Variable A(0,0) bis A(8,1).
3. Für ein mehrdimensionales Feld werden pro Index 11 Plätze reserviert.

Zur Dimensionierung größerer Felder steht der DIM-Befehl (in der Form DIM(15,12) zur Verfügung.

Als zweidimensionale Felder sehen unsere Bestenlisten inklusive der bisher noch nicht behandelten Tages-Bestenliste jetzt so aus:

	Ewige Liste	Persönliche Liste	Tages-Liste
Punkte	PL(K,0)	PL(K,1)	PL(K,2)
Namen	NA$(K,0)	NA$(K,1)	NA$(K,2)

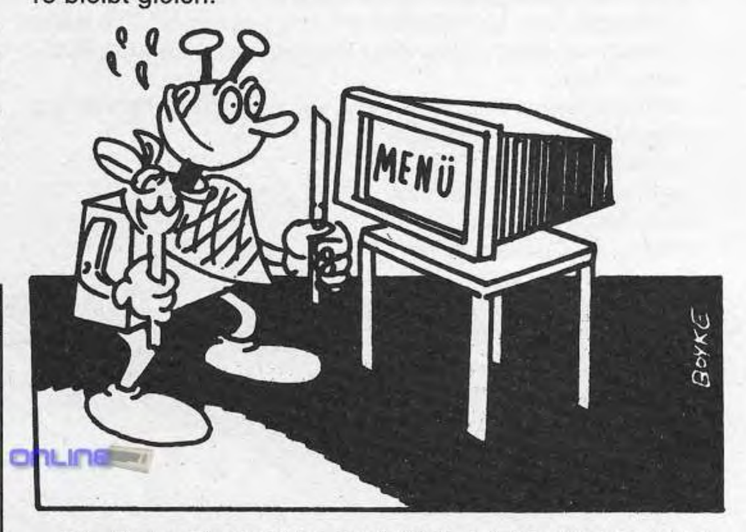
Sie sehen, die Namen der Variablen und der erste Index K, der bislang für die zehn Plätze in der Liste immer von 1 bis 10 gelaufen ist, sind gleich geblieben.

Neu ist der zweite Index, der den Listentyp kennzeichnet. Ist er 0, handelt es sich um die ewige Bestenliste, 1 oder 2 kennzeichnen die beiden anderen Listen. Der zweite Index erhält wie der erste einen Namen: ich nenne ihn F (für Flagge). Immer wenn die Flagge F auf einen der drei Werte 0,1 oder 2 gesetzt ist, ist festgelegt, um welche Bestenliste es sich im folgenden handelt.

Das Programm der persönlichen Bestenliste sieht so aus:

```
110 INPUT "NAME";SN$
120 INPUT "RESULTAT=";RE
130 F=0
140 NL$=SN$:PZ=RE
150 FOR K=1 TO 10
160 IF NA$=NA$(K,F) AND PZ <= PL(K,F) THEN PZ=0
170 IF NA$=NA$(K,F) AND PZ >PL(K,F) THEN PL(K,F)=
    0:NA$(K,F)=""
180 NEXT K
210 FOR K=1 TO 10
220 IF PZ >PL(K,F) THEN GOSUB 1100
225 PRINT K;PL(K,F);NA$(K,F)
230 NEXT K
240 GET A$: IF A$="" THEN 240
250 GOTO 110
1100 A=PL(K,F)
1110   PL(K,F)=PZ
1120            PZ=A
1130 B$=NA$(K,F)
1140    NA$(K,F)=NA$
1150             NA$=B$
1160 RETURN
```

Ich habe vorher schon betont, daß die ewige Bestenliste sich von dem obigen Programm nur dadurch unterscheidet, daß die Flagge in Zeile 130 auf F=1 stehen muß, und daß die Prüfung der Zeilen 150 bis 180 wegfällt. Alles andere bleibt gleich.



Die Tages-Bestenliste schließlich ist identisch mit der ewigen Bestenliste, nur wird sie nicht gespeichert.

Auffällig ist, wie ich meine, daß nicht nur das Unterprogramm ab Zeile 1100 in allen drei Listen vorkommt, sondern auch der Sortierblock der Zeilen 210 bis 250. Programmtechnisch bietet sich daher an, diesen Teil ebenfalls als Unterprogramm auszulegen. Ich siedle es ab Zeile 1000 an und versehe es mit der Rücksprungzeile 1030.

```
1030 RETURN
```

Nun ist nur noch ein Sprung auf dieses Unterprogramm erforderlich:

```
190 GOSUB 1000
```

Menü-Auswahl

Um die Verwendung der Bestenlisten so bequem wie möglich zu machen, muß es möglich sein, sie nach Belieben auszudrucken, nach einem Spiel zu speichern oder vor einem Spiel einzuladen. Dabei sollte die Bedienung einfach gehalten sein.

Dazu dient ein »Menü«. Wir wollen deshalb für die Bestenlisten ein solches Menü entwerfen, wobei ich vorschlage:

- NEUES SPIEL
- EWIGE BESTENLISTE DRUCKEN
- PERSÖNLICHE BESTENLISTE DRUCKEN
- TAGESBESTENLISTE DRUCKEN
- SPEICHERN DER LISTEN
- LADEN DER LISTEN
- ENDE

Der nun folgende Programmteil sorgt dafür, daß dieses Menü auf dem Bildschirm erscheint.

```
20 PRINT CHR$(147) CHR$(17) CHR$(17) TAB(10) "MENUE"
25 PRINT:PRINT:PRINT"N = NEUES SPIEL"
30 PRINT:PRINT       "E = EWIGE BESTENLISTE DRUCKEN"
35 PRINT:PRINT       "P = PERSOENLICHE BESTENLISTE
                          DRUCKEN"
40 PRINT:PRINT       "T = TAGESBESTENLISTE DRUCKEN"
45 PRINT:PRINT       "S = SPEICHERN DER LISTEN"
50 PRINT:PRINT       "L = LADEN DER LISTEN"
55 PRINT:PRINT       "Q = ENDE"
```

Hier ist eigentlich nur anzumerken, daß eine Leerzeile sowohl mit dem Befehl PRINT CHR$(17) als auch mit einem PRINT-Befehl ohne weitere Angaben erzielt werden kann. Beachten Sie auch bitte die Leerstellen innerhalb der Gänsefüße. Ohne sie würde der Text am Bildschirmrand kleben.

Zur Erinnerung

1. Durch den Befehl PRINT CHR$( ) wird das Zeichen ausgedruckt, dessen ASCII-Codezahl zwischen den Klammern steht.
2. Steht in der Klammer die ASCII-Codezahl eines Steuerzeichens (zum Beispiel CURSOR-UP), dann wird seine Funktion ausgeführt.
3. Eine Tabelle aller ASCII-Codezahlen ist in den Commodore-Handbüchern als Anhang enthalten (siehe auch Tabelle 2 in Teil 1 dieses Kurses).

Nach dem Ausdruck des Menüs auf dem Bildschirm soll nun der Benutzer eine der sieben Möglichkeiten durch Eintippen des vorgestellten Buchstabens auswählen.

Dazu fordern wir ihn in Zeile 60 auf und warten mit dem GET-Befehl:

```
60 PRINT SPC(180) "BITTE WAEHLEN"
70 GET A$: IF A$="" THEN 70
```

Diesmal habe ich den SPC-Befehl verwendet, um den Cursor vier Zeilen tiefer (160 Leerstellen) plus 20 Stellen nach rechts zu positionieren.

So, jetzt müssen wir abfragen, welcher der sieben Buchstaben eingegeben worden ist. Die einfachste Methode bietet der IF-THEN-Befehl:

```
 75 IF A$="N" THEN 110
 80 IF A$="E" THEN 400
 85 IF A$="P" THEN 500
 90 IF A$="T" THEN 600
 95 IF A$="S" THEN 700
100 IF A$="L" THEN 800
105 IF A$="Q" THEN 900
```

Die erste Sprungadresse kennen Sie aus unserem bisherigen Programm. Die Zeilennummern 400 bis 900 gibt es noch nicht; dort werden wir die Programmteile für »Drukken, Laden und Speichern« ansiedeln.

Die vielen IF-THEN-Abfragen lassen sich eleganter mit dem ON-GOTO-Befehl ersetzen. Sie sind mit dem ON-GOTO-Befehl nicht vertraut? Dann lesen Sie den nächsten »Erinnerungskasten«.

Zur Erinnerung

1. Der Befehl ON-GOTO wird so geschrieben: ON Variable GOTO mehrere Zeilennummern, wobei die Zeilennummern durch Kommata getrennt sein müssen.
2. Der Wert der Variablen, die hinter dem ON steht, legt fest, auf welche der hinter dem GOTO folgenden Zeilennummern verzweigt wird.
3. Entspricht der Variablenwert keiner ganzen Zahl (zum Beispiel 1,5), dann verzweigt der GOTO-Befehl auf die Zeilennummer, die entsteht, wenn man die Nachkommastellen abschneidet (also im Beispiel auf die Zeile mit der Nummer 1).

Der die IF-THEN-Abfragen ersetzende ON-GOTO-Befehl sieht folglich so aus:

```
95 ON X GOTO 110,400,500,600,700,800,900
```

Die Sprungadressen hinter dem GOTO sind also wieder identisch mit den Sprungadressen der IF-THEN-Zeilen.

Vor dieser Zeile 95 müssen jetzt für X die Zahlen 1, 2, 3, 4, 5, 6 oder 7 erzeugt werden, je nachdem, welcher der sieben Buchstaben des Menüs gewählt worden ist.

Das bewerkstelligen wir mit einem, wie ich finde, sehr eleganten String-Trick.

String-Hilfe für ON-GOTO

Ich schreibe den Programmteil erst einmal hin und erkläre ihn anschließend:

```
70 GET A$:IF A$="" THEN 70
75 FOR X=1 TO 7
80 IF A$=MID$("NEPTSLQ",X,1) THEN 95
85 NEXT X
90 GOTO 20
95 ON X GOTO 110,400,500,
600,700,800,900
```

Die Zeile 70 haben wir schon gehabt. Sie wartet auf die Eingabe eines der sieben Buchstaben des Menüs. Auch die Zeile 95 mit dem ON-GOTO-Befehl habe ich schon beschrieben.

Der eigentliche Pfiff liegt in der Abfrage-Zeile 80, die durch die X-Schleife siebenmal durchlaufen wird.

Bei jedem Durchlauf wird mit dem MID$-Befehl aus dem künstlichen String, den wir aus den sieben Anfangsbuchstaben des Menüs »NEPTSLQ« gebildet haben, jeder Buchstabe einzeln herausgeschnitten.

Zur Erinnerung

1. Der String-Befehl MID$ (X$,B,A) schneidet vom String X$ von links her ab dem B-ten Zeichen insgesamt A Zeichen heraus und bildet daraus einen neuen String.
2. MID$("MENUE",3,2) erzeugt demnach den neuen String »NU«.
3. Bei MID$ kann die zweite Zahl A weggelassen werden. Dann wird ab dem B-ten Zeichen der Rest des Strings X$ herausgeschnitten.

Tritt eine Übereinstimmung mit dem eingetippten Buchstaben A$ auf, gilt der zugehörige Wert der Schleifenvariablen X als Variablenwert für den ON-GOTO-Sprung.

Dieser letzte Satz klingt derartig theoretisch, daß ich ihn mit einem Beispiel verständlich machen will.

Angenommen, Sie wollen die Tages-Bestenliste ausdrucken und wählen daher den Buchstaben T. Beim ersten Durchgang der Schleife (X=1) schneidet der MID$-Befehl ab dem X-ten, also ab dem ersten Zeichen des Strings »NEPTSLQ« ein Zeichen heraus – es ist das N – und vergleicht es mit dem eingegebenen Buchstaben T. Wir sehen, daß erst beim vierten Durchlauf, also bei X=4, der herausgeschnittene Buchstaben identisch mit dem vorher ausgewählten Buchstaben T ist. Das Programm verzweigt durch Erfüllung der Abfragebedingung aus der Zeile 80 in die Zeile 95 und verwendet dort den Wert 4 für X. Dadurch springt der ON-GOTO-Befehl auf die vierte Zeilennummer hinter dem GOTO. Das ist die Zeile 600, in der in der Tat der Programmteil für das Ausdrucken der Tages-Bestenliste beginnen soll.

Ausdruck der Listen

Bisher haben wir alle drei Bestenlisten gleich beim Einsortieren eines neuen Ergebnisses ausgedruckt, und zwar mit Hilfe der Zeile 225, die später im Unterprogramm zur Zeile 1015 wurde.

Von dort nehmen wir sie jetzt heraus, weil wir erstens das Ausdrucken über das Menü als Unterprogramm anwählen

wollen und weil zweitens das Format der Listen bislang äußerst primitiv war.

Unter einer »schönen« Liste stelle ich mir folgendes Format vor:

```
 1. BEATE         765
 2. FRANZ BERGH    98
 3. XAVER          62
    und so weiter
 9. HELENE         12
10. FRANZ MAIER     9
```

Dieses Listenformat stellt bestimmte Anforderungen:

1. Die Platzzahlen links sind nach dem Punkt ausgerichtet,
2. die Ergebnisse sind nach dem rechten Rand ausgerichtet,
3. die Länge der Spielernamen ist begrenzt auf zwei Stellen vor dem größten Ergebnis (das natürlich immer als erste Zahl in einer Ergebnisliste steht).

Das werden wir jetzt programmieren.

1. Forderung: Zuerst wird die Positionsnummer mit nachfolgendem Punkt ausgedruckt:

```
1210 FOR K=1 TO 10
1230 PRINT TAB(2) K;
1240 PRINT ". ";
1280 NEXT K
```

Mit diesen Zeilen würde die Zahl K (1 bis 10) zwei Stellen vom linken Rand entfernt geschrieben werden. Um aber die Zahlen auf den Punkt auszurichten, verwenden wir folgenden String-Trick:

```
1220 J=1:IF K>9 THEN J=2
1230 PRINT TAB(2) MID$(STR$(K),J);
```

In Zeile 1230 wandeln wir mit dem STR$-Befehl die Zahl K in einen String um, wobei der STR$-Befehl die Vorzeichenstelle beibehält. STR$(5) ergibt demnach einen zwei Zeichen umfassenden String mit, STR$(15) einen mit drei Zeichen.

Der MID$-Befehl vor dem STR$-Teil schneidet diesen neuen String ab der Stelle ab, der durch J angegeben ist. Aus Zeile 1220 sehen wir, daß J für die Werte von K=1 bis 9 gleich 1 ist ; ab K=10 ist J aber 2.

Mit anderen Worten: die Zahlen 1 bis 9 werden mit ihrer (leeren) Vorzeichenstelle gedruckt, die Zahlen 10 bis 99 ohne diese Vorzeichenstelle. Beide sind also immer zwei Stellen lang. Dadurch kommt der Punkt, den Zeile 1240 druckt, immer an dieselbe Stelle.

Vorsicht: Der Trick geht natürlich nicht mehr bei dreistelligen Zahlen.

2. Forderung: Wenn die Ergebnisse PL(K,F) mit einem TAB(20) versehen ausgedruckt würden, stünden sie mit ihrem linken Rand ausgerichtet auf dem Bildschirm. Die rechte Ausrichtung erzielen wir dadurch, daß wir von dem TAB-Wert, der den rechten Rand darstellen soll, die jeweilige Länge des Ergebnisses abziehen. Dazu muß, so wie vorher, die Zahl PL(K,F) in einen String umgewandelt werden – STR$(PL(K,F)) –.

Davon errechnen wir die Länge mit – LEN(STR$(PL(K,F))).

Der komplette TAB-Befehl sieht dann so aus: – TAB(20-LEN(STR$(PL(K,F)))) –.

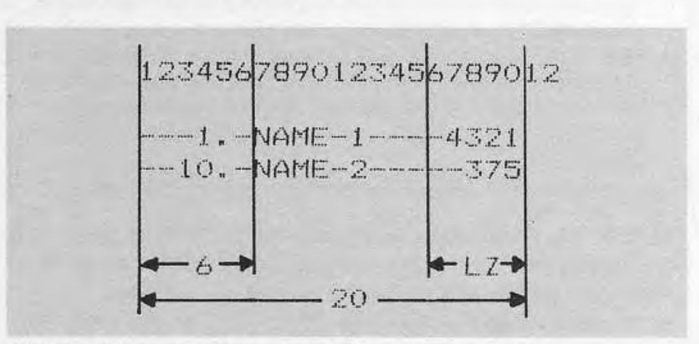


Bild 25. Die formatierte Ausgabe der Bestenliste

Beachten Sie bitte die Klammern! Es müssen immer so viele Klammern geschlossen werden, wie vorher geöffnet worden sind – in unserem Fall sind es vier!

```
1270 PRINT TAB(20-LEN(STR$(PL(K,F)))); PL(K,F)
```

Erst nach dem TAB-Ausdruck kommt der eigentliche PRINT-Befehl.

3. Forderung: Wir wollen vermeiden, daß allzu lange Namen das Ergebnis von dem vorher so sorgfältig angeordneten Platz verdrängen. Wie das geht, zeigt uns Bild 25.

Links wird der Name durch die Positionszahl begrenzt; in unserem Beispiel beginnt er immer ab der siebten Stelle.

Rechts ist er begrenzt vom Rand bei TAB(20), abzüglich der Länge LZ des größten Ergebnisses, welches natürlich immer an erster Position der Liste steht. Demnach stehen für den Namen maximal 20-6-LZ Stellen (siehe Bild 25) zur Verfügung.

Die Länge LZ bestimmen wir mit derselben Methode wie vorher in Zeile 1250. Den ersten Index fixieren wir auf 1 (für die erste Position), der zweite bleibt wie gehabt auf der Flagge F.

```
1250 LZ=LEN(STR$(PL(1,F)))
```

Mit dem LEFT$-Befehl schneiden wir von links her vom Namen NL$(K,F) die oben ausgerechnete maximale Länge (20-6-LZ) ab und drucken ihn aus:

```
1260 PRINT LEFT$(NL$(K,F),14-LZ);
```

Den Abschluß bildet eine Wartezeile und ein RETURN-Befehl:

```
1290 GET A$:IF A$="" THEN 1290
1295 RETURN
```

Die Zeile 1290 läßt uns in aller Ruhe das Werk genießen.

Die Zeile 1295 wird notwendig, da ich diesen Programmteil, der ja beim Ausdrucken jeder Liste vorkommt, ebenfalls als Unterprogramm vorsehe. Wichtig ist dann nur, daß in den Abschnitten ab 400, 500 und 600 zuerst die richtige Flagge F gesetzt und dann mit GOSUB 1200 auf das Druck-Unterprogramm gesprungen wird.

Diese ganze Anordnung können Sie im endgültigen Listing 27 wiederfinden.

```
100 REM** PERSOENLICHE BESTENLISTE **
102 :                                          <078>
105 REM--- PERSOENLICHE BESTENLISTE ---        <223>
107 :                                          <083>
110 INPUT "NAME";NL$                           <011>
120 INPUT"RESULTAT =";PZ                       <236>
130 :                                          <106>
150 FOR K=1 TO 10                              <056>
160 IF NL$=NL$(K) AND PZ<=PL(K) THEN PZ=0      <195>
170 IF NL$=NL$(K) AND PZ>PL(K) THEN PL(K)=
    0:NL$(K)=""                                <111>
180 NEXT K                                     <024>
199 :                                          <175>
210 FOR K=1 TO 10                              <116>
220 IF PZ>PL(K) THEN GOSUB 1100                <210>
225 PRINT K;PL(K);NL$(K)                       <160>
230 NEXT K                                     <074>
240 GET A$:IF A$="" THEN 240                   <146>
250 GOTO 110                                   <194>
299 :                                          <021>
1100 A=PL(K)                                   <084>
1110 PL(K)=PZ                                  <147>
1120 PZ=A                                      <206>
1125 :                                         <085>
1130 B$=NL$(K)                                 <023>
1140 NL$(K)=NL$                                <100>
1150 NL$=B$                                    <091>
1160 RETURN                                    <202>
```

© 64'er

Listing 26. Die persönliche Bestenliste erzeugt dieser Programmteil, der sich auch in eigenen Programmen verwenden läßt

```
5 REM******************************          <059>
6 REM******* BESTENLISTEN *********          <116>
7 REM******************************          <061>
8 :                                          <240>
10 DIM PL(60,2),NL$(60,2)                    <156>
14 :                                         <246>
15 REM---------- MENUE -------------         <095>
16 :                                         <248>
20 PRINT CHR$(147) CHR$(17) CHR$(17) TAB(1
   0)"MENUE"                                 <172>
25 PRINT:PRINT:PRINT "{2SPACE}N = NEUES SP
   IEL"                                      <199>
30 PRINT:PRINT "{2SPACE}E = EWIGE BESTENLI
   STE"                                      <089>
35 PRINT:PRINT "{2SPACE}P = PERSOENLICHE B
   ESTENLISTE"                               <092>
40 PRINT:PRINT "{2SPACE}T = TAGESBESTENLIS
   TE"                                       <211>
45 PRINT:PRINT "{2SPACE}S = SPEICHERN DER
   LISTEN"                                   <141>
50 PRINT:PRINT "{2SPACE}L = LADEN DER LIST
   EN"                                       <088>
55 PRINT:PRINT "{2SPACE}Q = ENDE"            <123>
60 PRINT SPC(180) "BITTE WAEHLEN"            <184>
65 PRINT:PRINT                               <017>
70 GET A$:IF A$="" THEN 70                   <111>
75 FOR X=1 TO 7                              <040>
80 IF A$=MID$("NEPTSLQ",X,1) THEN 95         <211>
85 NEXT X                                    <033>
90 GOTO 20                                   <020>
95 ON X GOTO 110,400,500,600,700,800,900     <121>
99 :                                         <075>
100 REM----- NEUES SPIEL ------              <099>
102 :                                        <078>
110 INPUT "NAME";SN$                         <047>
120 INPUT"RESULTAT =";RE                     <216>
125 :                                        <101>
130 F=0:REM----- PERSOENLICHE LISTE-----     <211>
133 :                                        <109>
140 NL$=SN$:PZ=RE                            <043>
150 FOR K=1 TO 60                            <187>
160 IF NL$=NL$(K,F) AND PZ<=PL(K,F) THEN P
    Z=0                                      <229>
170 IF NL$=NL$(K,F) AND PZ>PL(K,F) THEN PL
    (K,F)=0:NL$(K,F)=""                      <153>
180 NEXT K                                   <024>
190 GOSUB 1000                               <146>
199 :                                        <175>
230 F=1:REM----- EWIGE BESTEN-LISTE-----     <250>
233 :                                        <209>
240 NL$=SN$:PZ=RE                            <143>
250 GOSUB 1000                               <206>
299 :                                        <021>
330 F=2:REM----- TAGES-BESTEN-LISTE-----     <005>
333 :                                        <055>
340 NL$=SN$:PZ=RE                            <245>
350 GOSUB 1000                               <052>
360 GOTO 20                                  <036>
399 :                                        <121>
400 REM--- DRUCKEN DER EWIGEN LISTE ---      <024>
405 :                                        <127>
410 F=1                                      <093>
420 PRINT CHR$(147):PRINT                    <157>
430 PRINT TAB(3)"EWIGE BESTENLISTE"          <044>
440 PRINT                                    <032>
450 GOSUB 1200                               <184>
460 GOTO 20                                  <136>
499 :                                        <221>
500 REM--- DRUCKEN DER PERSON.LISTE ---      <189>
505 :                                        <227>
510 F=0                                      <177>
520 PRINT CHR$(147):PRINT                    <003>
530 PRINT TAB(3)"PERSOENL. LISTE"            <253>
540 PRINT                                    <134>
550 GOSUB 1200                               <030>
560 GOTO 20                                  <238>
599 :                                        <067>
600 REM--- DRUCKEN DER TAGES  LISTE ---      <111>
605 :                                        <073>
610 F=2                                      <055>
620 PRINT CHR$(147):PRINT                    <103>
630 PRINT TAB(3)"TAGES-BESTENLISTE"          <233>
640 PRINT                                    <234>
650 GOSUB 1200                               <130>
660 GOTO 20                                  <082>
699 :                                        <167>
700 REM----- SPEICHERN DER LISTE ------      <174>
705 :                                        <173>
710 OPEN 1,8,3,"TAB.,S,W"                    <008>
720 FOR K=1 TO 60                            <249>
730 IF NL$(K,0)=""THEN NL$(K,0)="."          <175>
740 IF NL$(K,1)=""THEN NL$(K,1)="."          <219>
750 PRINT#1,NL$(K,0):PRINT#1,NL$(K,1)        <228>
760 PRINT#1,STR$(PL(K,0)):PRINT#1,STR$(PL(
    K,1))                                    <241>
770 NEXT K                                   <108>
780 CLOSE 1                                  <029>
790 GOTO 20                                  <214>
799 :                                        <013>
800 REM----- LADEN DER LISTEN --------       <103>
805 :                                        <019>
810 OPEN 1,8,4,"TAB.,S,R"                    <229>
820 FOR K=1 TO 60                            <095>
830 INPUT#1,NL$(K,0):IF NL$(K,0)="."THEN N
    L$(K,0)=""                               <201>
840 INPUT#1,NL$(K,1):IF NL$(K,1)="."THEN N
    L$(K,1)=""                               <255>
850 INPUT#1,PL$(0),PL$(1)                    <159>
860 PL(K,0)=VAL(PL$(0))                      <202>
870 PL(K,1)=VAL(PL$(1))                      <085>
880 NEXT K                                   <218>
890 CLOSE 1                                  <139>
895 GOTO 20                                  <063>
899 :                                        <113>
900 REM------SPIEL-ENDE --------             <152>
905 :                                        <119>
910 END                                      <150>
994 REM******************************        <026>
995 :                                        <209>
996 REM******* UNTERPROGRAMME *********      <101>
997 :                                        <211>
998 REM------- SORTIEREN 1 ---------         <072>
999 :                                        <213>
1000 FOR K=1 TO 60                           <019>
1010 IF PZ>PL(K,F) THEN GOSUB 1100           <047>
1020 NEXT K                                  <102>
1030 RETURN                                  <072>
1097 :                                       <057>
1098 REM------- SORTIEREN 2 ---------        <182>
1099 :                                       <059>
1100 A=PL(K,F)                               <135>
1110 PL(K,F)=PZ                              <187>
1120 PZ=A                                    <206>
1125 :                                       <085>
1130 B$=NL$(K,F)                             <231>
1140 NL$(K,F)=NL$                            <098>
1150 NL$=B$                                  <091>
1160 RETURN                                  <202>
1199 :                                       <159>
1200 REM---- DRUCKEN DER LISTEN -----        <217>
1205 :                                       <165>
1210 FOR K=1 TO 60                           <231>
1215 IF K/21=INT(K/21) THEN GET A$:IF A$="
     " THEN 1215                             <002>
1220 J=1: IF K>9 THEN J=2                    <239>
1230 PRINT TAB(2) MID$(STR$(K),J);           <227>
1240 PRINT". ";                              <075>
1250 LZ=LEN(STR$(PL(1,F)))                   <110>
1260 PRINT LEFT$(NL$(K,F),14-LZ);            <213>
1270 PRINT TAB(20-LEN(STR$(PL(K,F))));PL(K
     ,F)                                     <227>
1280 NEXT K                                  <110>
1290 GET A$:IF A$=""THEN 1290                <201>
1295 RETURN                                  <083>
```


Listing 27. Das komplette Programm zur Erstellung aller in diesem Kursteil besprochenen Bestenlisten

Zur Erinnerung

Der Befehl STR$(X) bildet aus dem Wert der Zahl X einen String. Dieser String behält die Vorzeichenstelle von X als Leerstelle.

Jetzt fehlen nur noch die Programmteile für das Laden und Speichern der ewigen Bestenliste und der persönlichen Bestenliste. Wie schon gesagt, wird die Tagesbestenliste nicht gespeichert.

Das Prinzip, eine Liste auf Band oder Diskette zu speichern, gehört eigentlich nicht in einen String-Kurs. Es ist übrigens auch in Kurzform im Basic-Kurs des Sonderheftes 40 (ab Seite 98) beschrieben worden.

Ich gehe hier nur auf die String-Befehle näher ein.

Speichern einer Liste

Unsere Listen werden als »sequentielle Datei« unter dem freigewählten Namen »TAB.« gespeichert.

```
710 OPEN 1,8,3,"TAB.,S,W" (auf Diskette)
710 OPEN 1,1,1,"TAB."     (auf Kassette)
720 FOR K=1 TO 10
750 PRINT#1,NL$(K,0):PRINT#1,NL$(K,1)
```

Nach dem OPEN-Befehl werden die Namen mit dem PRINT#-Befehl gespeichert, wobei die beiden Listentypen (F=0,F=1) getrennt behandelt werden.

Ein Problem kann dabei auftreten, wenn ein Name eine Leerstelle enthält (zum Beispiel Franz Obermaier). Diese Leerstellen müssen künstlich mit einem Zeichen gefüllt werden, das dann später beim Laden wieder zu entfernen ist.

```
730 IF NL$(K,0)="" THEN NL$(K,0)="."
740 IF NL$(K,1)="" THEN NL$(K,1)="."
```

Eine zweite Forderung ist, daß auch die Zahlen PL(K,F) als Strings gespeichert werden müssen. Wie das geht, haben wir schon vorher erarbeitet:

```
760 PRINT#1,STR$(PL(K,0)):PRINT#1,STR$(PL(K,1))
770 NEXT K
780 CLOSE 1
790 GOTO 20
```

In Zeile 760 setzen wir den STR$-Befehl wieder ein. Zeile 770 beschließt die Schleife, Zeile 790 verzweigt zurück in das Menü.

Auf Band oder Diskette gespeicherte Listen lassen sich mit der umgekehrten Reihenfolge des Speicherns wieder in ein Spiel holen.

```
810 OPEN 1,8,4,"TAB.",S,R" (für Diskette)
810 OPEN 1,1,0,"TAB."      (für Kassette)
820 FOR K=1 TO 10
```

Nach dem OPEN-Befehl werden die gespeicherten Strings in derselben Reihenfolge wie beim Speichern mit dem INPUT#-Befehl eingelesen.

```
830 INPUT#1,NL$(K,0)
840 INPUT#1,NL$(K,1)
```

Dabei müssen aber die in Punkte umgewandelten Leerstellen wieder zurückgewandelt werden. Die beiden Zeilen werden daher erweitert zu:

```
830 INPUT#1,NL$(K,0): IF NL$(K,0)="."
    THEN NL$(K,0)=""
840 INPUT#1,NL$(K,1): IF NL$(K,1)="."
    THEN NL$(K,1)=""
```

Dann kommen die in Strings verwandelten Zahlen unter irgendeinem Namen an die Reihe. Ich nehme einfach PL$(0) und PL$(1).

```
850 INPUT#1,PL$(0),PL$(1)
```

Diese Strings werden jetzt mit dem VAL-Befehl in Zahlen zurückverwandelt.

```
860 PL(K,0)=VAL(PL$(0))
870 PL(K,1)=VAL(PL$(1))
```

Der Rest ist identisch mit dem Speichern:

```
880 NEXT K
890 CLOSE 1
895 GOTO 20
```

Zur Erinnerung

Der Befehl VAL(A$) liefert den numerischen Wert des Strings A$. Dieser String kann sowohl mit dem $-Zeichen als auch in Gänsefüßchen geschrieben sein.

Das war eigentlich alles, was dieses Programm an String-Verarbeitung zu bieten hat. Mir fällt nur noch eine kleine Verbesserung ein, die zwar nichts mit Strings zu tun hat, aber recht nützlich sein kann.

Ich schlage vor, die zehn Eintragungen der Listen auf 60 zu erhöhen, das sind gerade drei mal zwanzig. Warum 3 x 20? Nun, weil der Bildschirm nicht alle 60 Eintragungen auf einmal darstellen kann und wir deshalb »umblättern« müssen.

Doch zuerst muß ich Sie an etwas erinnern, was ich ganz am Anfang gesagt habe: Ein Feld mit mehr als elf Plätzen muß vorher dimensioniert werden.

```
10 DIM PL(60,2),NL$(60,2)
```

Der zweite Index ist wieder das F beziehungsweise die maximale Anzahl der F-Werte. Dann müssen wir alle K-Schleifen auf 60 erweitern.

```
 150 FOR K=1 TO 60
 720 FOR K=1 TO 60
 820 FOR K=1 TO 60
1000 FOR K=1 TO 60
1210 FOR K=1 TO 60
```


Wenn Sie das Programm jetzt laufen lassen, sausen die Anfänge der Tabellen aus dem Bildschirm.

Drucken mit Pausen

Nach jeweils 20 Zeilen soll das Ausdrucken aber stehenbleiben, so lange, bis eine Taste gedrückt wird. Dazu verwenden wir eine kleine mathematische Formel:

```
FOR K=1 TO 60
IF  K/21 = INT(K/21) THEN...
```

Während das Programm die Druckschleife der Zeilen 1210 bis 1280 bearbeitet, wird jedesmal der Wert der Schleifenvariablen K durch 21 dividiert. Nur wenn das Resultat eine ganze Zahl ist, was wir mit dem INT-Befehl prüfen, dann soll das Programm warten.

Diese Bedingung ist nur bei den Werten K=21 und K=42 erfüllt, denn 21/21=1 und 42/21=2.

Hinter dieser Abfrage warten wir wie üblich mit dem GET-Befehl. Dadurch werden alle drei Listen beim Ausdrucken in drei lesbare Teile geteilt.

```
1210 FOR K=1 TO 60
1215 IF K/21=INT(K/21) THEN GET A$: IF A$=""
     THEN 1215
```

Jetzt können Sie mit irgendeiner Taste die Listen bequem umblättern.

Damit ist das Ende des String-Kurses erreicht, und Sie können sich nun zu der Gruppe der fortgeschrittenen Basic-Programmierer zählen. Die vielfältigen Beispiele des Kurses haben nur angedeutet, welche Vielfalt der unterschiedlichsten Anwendungsmöglichkeiten mit der Kenntnis der Stringverarbeitung vorhanden ist. Vielleicht finden Sie bei Ihren weiteren Erkundungen durch die Welt der Strings geeignete Lösungen oder interessante Anwendungen, die wir in späteren Heften vorstellen können.

(Dr. Helmuth Hauck/Dr. R. Egg/ef)

Menügesteuert laden

In Zukunft laden Sie von Ihren Disketten nur noch ein einziges Programm: das Menü. Danach wählen Sie das gewünschte Programm direkt an und lassen es laden.

Komfortables Laden von der Diskette – ohne langes Suchen nach dem Lade-Programm. Keine Frage mehr, ob das Programm mit »,8« oder mit »,8,1« geladen wird. Das Menü übernimmt diese Kleinigkeiten für Sie. Dabei benötigt es nur etwa sechs bis sieben Blöcke auf der Diskette (je nach Länge des Directory) und dürfte damit auf fast jeder Diskette Platz haben. Das Programm, das dieses Menü für Sie erstellt, ist der »MENUE-MAKER« (Listing 1).

Bedienungsanleitung:

Sie laden den Menü-Maker und starten ihn mit RUN. Dann legen Sie die Diskette ein, für die Sie ein Menü erstellen wollen (ein eventueller Schreibschutz ist zu entfernen), und wählen Menüpunkt 2.

Das Programm lädt nun das Directory und zeigt das erste Programm auf der Diskette an. Wenn das Programm später mit »,8« geladen werden muß, dann drücken Sie <F1>, wenn Sie es mit »,8,1« laden müssen, dann drücken Sie <F3>. Wenn es sich um ein Unterprogramm oder sonst ein Programm handelt, das für sich selber nicht lauffähig ist, oder von einem Lader nachgeladen wird, dann drücken Sie <F5> (das Programm wird nicht ins Menü übernommen).

Wenn Sie so alle Programme auf der Diskette durchgegangen sind, erscheint wieder das Hauptmenü. Wählen Sie nun Punkt 4 und das Disk-Menü wird zusammen mit der Datei auf Diskette gespeichert. Sollte Ihnen bei Punkt 2 ein Fehler unterlaufen sein, dann wählen Sie diesen nochmals an und gehen ihn wieder durch, bis alles richtig ist. Dann fahren Sie weiter, wie oben beschrieben.

Zusatzfunktionen: Wenn Sie Menüpunkt 1 anwählen, können Sie sich das Directory anschauen. Diese Funktion hat keinen Einfluß auf Punkt 2 und 4.

Durch Wählen von Punkt 3 erhalten Sie eine kurze Anleitung. Mit Punkt 5 können Sie das Programm beenden. Zur Sicherheit werden Sie noch einmal gefragt, ob Sie das Programm wirklich verlassen wollen. Wenn ja, wird ein Reset ausgeführt. Im anderen Fall sind Sie wieder im Hauptmenü. Punkt 1 und 3 verlassen Sie durch Drücken einer beliebigen Taste.

Programmbeschreibung:

Die wesentlichen Aspekte des Programms sind Menüpunkt 2 und 4.

Zu 2: Aus dem Directory werden die Programmnamen eingelesen und der indizierten Variablen NA$ (IN) zugeordnet. Dann wartet das Programm, bis man eine der drei möglichen Funktionstasten gedrückt hat. Wenn man <F1> gedrückt hat, wird NU$ (IN) gleich 8 gesetzt, damit das Programm später weiß, daß es dieses Programm mit »,8« laden muß. Entsprechend wird nach Drücken von <F3> NU$ (IN) auf 81 gesetzt. Wird <F5> gedrückt, so wird NA$ (IN) wieder gelöscht und der Zähler IN um 1 zurückgesetzt. Diese Prozedur wird fortgesetzt, bis alle Programme im Directory abgearbeitet sind und das Programm wieder zum Menü springt.

Zu 4: Hier werden nun zuerst NA$ (IN) (das den Programmnamen enthält) und NU$ (IN) (das die Information enthält, ob mit »,8« oder mit »,8,1« geladen werden soll) als sequentielle Datei mit dem Namen »MSD« gespeichert. Ist dies geschehen, wird der Bildschirm gelöscht und folgendes darauf geschrieben (was man jedoch nicht sieht, da es in der Hintergrundfarbe geschrieben wird):

```
RUN 1600 (3 Zeilen Abstand)
POKE 43,PEEK (61)+1: POKE 44, PEEK (62)
(2 Zeilen Abstand)
SAVE "MENU",8 (4 Zeilen Abstand)
POKE 43,1: POKE 44,8 (2 Zeilen Abstand)
GOTO 1260
```

Dann wird der Tastaturpuffer mit RETURN gefüllt, und durch einen END-Befehl fährt das Programm im Direktmodus fort.

Auf dem Bildschirm stehen nun jedoch die oben genannten Befehle; der Tastaturpuffer ist mit RETURN gefüllt. Also führt der Computer diese Befehle aus.

Durch die ersten beiden Befehle wird das ab Zeile 1610 stehende Menü vom Rest des Programms getrennt (Anti-MERGE-Routine) und durch den SAVE-Befehl gespeichert.

Ist dies geschehen, so gelangt der Computer zu den beiden nächsten POKEs, die diese AntiMERGE-Routine wieder aufheben. Danach wird in einer neuen Zeile durch GOTO 1260 wieder ins Menü des Hauptprogrammes eingestiegen.

Damit wäre der Zweck dieses einfachen Programms schon erreicht, nämlich möglichst leicht und schnell ein Menü zu erstellen. Man muß also nur noch das Menü laden und mit RUN starten. Nachdem die Datei nachgeladen worden ist, kann man mit der Cursortaste und Return das zu ladende Programm wählen.

Das Menü lädt das Programm ebenfalls in einem »simulierten« Direktmodus nach. Es wird einfach ein LOAD- und ein RUN-Befehl auf den Bildschirm geschrieben, der Tastaturpuffer mit RETURN gefüllt und das Menü mit NEW gelöscht (wobei zugleich auch in den Direktmodus gesprungen wird). Übrigens kann das Menü (ab Zeile 1610) den eigenen Vorstellungen angepaßt werden. Man muß nur darauf achten, daß es nach dem STOP-Befehl in Zeile 1600 steht, da sonst die AntiMERGE-Routine das Menü nicht sauber abtrennt. (S. Brülisauer/kn)

Kurzinfo: Menü-Maker

Programmart: Floppy-Tool
Laden mit: LOAD "MENUE-MAKER",8
Starten mit: Nach dem Laden RUN eingeben
Eingaben über: Tastatur
Besonderheiten:
- Mit dem Programm können Sie ein komfortables Menü für eine Diskette erzeugen.
- Da die Programme aus dem Menü automatisch gestartet werden, ist beim Erzeugen der Menü-Datei darauf zu achten, ob das jeweilige Programm mit »,8« oder »,8,1« geladen werden muß.
- Das »MENU« selbst muß unbedingt mit »,8« und nicht mit »,8,1« geladen werden.

Programmautor: S. Brülisauer

```
10 DIM NA$(144),NU$(144)                                  <206>
20 PRINT CHR$(142):PRINT CHR$(8)                          <154>
30 GOTO 1020                                              <034>
40 REM *** ANLEITUNG ***                                  <052>
50 PRINT"{CLR}";:PRINT                                    <170>
60 PRINT SPC(14)"{BLUE,RVSON}ANLEITUNG                    <123>
70 PRINT                                                  <172>
80 PRINT"{WHITE}- DIESES PROGRAMM STELLT E
   IN MENU FUER                                           <086>
90 PRINT"{2SPACE}IHRE DISKETTEN HER UND SP
   EICHERT ES                                             <001>
100 PRINT"{2SPACE}AB. SIE MUESSEN ALSO NAC
    HHER NUR NOCH                                         <088>
110 PRINT"{2SPACE}DAS MENU LADEN,DAS PROGR
    AMM DAS SIE                                           <190>
120 PRINT"{2SPACE}WUENSCHEN AUSWAEHLEN UND
     SCHON WIRD                                           <115>
130 PRINT"{2SPACE}ES IN DEN RECHNER GELADE
    N.                                                    <055>
140 PRINT                                                 <242>
150 PRINT SPC(11)"{BLUE,RVSON}VORGEHENSWEI
    SE                                                    <056>
160 PRINT                                                 <006>
170 PRINT"{WHITE}- WAEHLEN SIE ZUERST MENU
    PUNKT 2 UND                                           <188>
180 PRINT"{2SPACE}BESTIMMEN SIE WELCHE PRO
    GRAMME INS                                            <164>
190 PRINT"{2SPACE}MENU AUFGENOMMEN WERDEN
    SOLLEN.                                               <134>
200 PRINT"{2SPACE}GLEICHZEITIG LEGEN SIE A
    UCH FEST,OB                                           <032>
210 PRINT"{2SPACE}ES MIT ,8 ODER MIT ,8,1
    (MASCHINEN-                                           <092>
220 PRINT"{2SPACE}PROGRAMME) GELADEN WERDE
    N SOLL.                                               <016>
230 PRINT"- WENN SIE DIESE ARBEIT ERLEDIGT
     HABEN,                                               <117>
240 PRINT"{2SPACE}SIND SIE AUTOMATISCH WIE
    DER IM HAUPT-                                         <070>
250 PRINT"{2SPACE}MENU UND KOENNEN NUN MEN
    UPUNKT 4 WAEH-";                                      <053>
260 PRINT"{2SPACE}LEN UM DAS MENU ABZUSPEI
    CHERN ODER                                            <171>
270 PRINT"{2SPACE}FALLS IHNEN BEI PUNKT 2
    EIN FEHLER                                            <222>
280 PRINT"{2SPACE}UNTERLAUFEN IST,NOCHMALS
     VON VORNE                                            <071>
290 PRINT"{2SPACE}BEGINNEN.{UP}"                          <233>
300 GET T$:IF T$=""THEN 300                               <017>
310 RETURN                                                <114>
320 :                                                     <042>
330 :                                                     <052>
340 REM *** UEBERNEHMEN ***                               <055>
350 PRINT"{CLR}";:A$="":T$="":IN=0:X=0:XX=
    0                                                     <037>
360 PRINT"{RVSON}F1 = ,8{RVOFF,SPACE,RVSON
    }F3 = ,8,1{RVOFF,SPACE,RVSON}F5 = NICH
    T UEBERNEHMEN{RVSON,DOWN}"                            <082>
370 CLOSE 2:OPEN 2,8,15                                   <188>
380 OPEN 1,8,0,"$0"                                       <020>
390 GET#1,A$,B$                                           <097>
400 GET#1,A$,B$                                           <107>
410 GET#1,A$,B$                                           <117>.
420 C=0                                                   <075>
430 IF A$<>"" THEN C=ASC(A$)                              <044>
440 IF B$<>"" THEN C=C+ASC(B$)*256                        <121>
450 GET#1,B$:IF ST<>0 THEN 540                            <138>
460 IF B$<>CHR$(34)THEN GOTO 450                          <062>
470 GET#1,B$:IF B$<>CHR$(34)AND B$<>"<"THE
    N A$=A$+B$:GOTO 550                                   <155>
480 IF B$<>CHR$(34)THEN GOTO 470                          <083>
490 IF X=0 THEN X=1:A$="":GOTO 520                        <014>
500 A$=RIGHT$(A$,LEN(A$)-1)                               <048>
510 A$=LEFT$(A$,LEN(A$)):PRINT A$;:GOSUB 5
    70                                                    <119>
520 GET#1,B$:IF B$=CHR$(32) THEN 520                      <026>
530 IF ST=0 THEN 400                                      <172>
540 CLOSE 1:PRINT"{CLR}";:RETURN                          <070>
550 IF LEN(A$)>20 THEN GOTO 540                           <050>
560 GOTO 470                                              <116>
570 IN=IN+1:NA$(IN)=A$                                    <109>
580 GET T$:IF T$=""THEN 580                               <109>
590 IF T$="{F1}"THEN NU$(IN)="8":POKE 214,
    PEEK(214):POKE 211,20:SYS 58640:PRINT"
    {RVSON},8{RVOFF}"                                     <004>
600 IF T$="{F1}"THEN RETURN                               <185>
610 IF T$="{F3}"THEN NU$(IN)="81":POKE 214
    ,PEEK(214):POKE 211,20:SYS 58640:PRINT
    "{RVSON},8,1{RVOFF}"                                  <085>
620 IF T$="{F3}"THEN RETURN                               <078>
630 IF T$="{F5}"THEN NA$(IN)="":IN=IN-1:PO
    KE 214,PEEK(214):POKE 211,20:SYS 58640                <014>
640 IF T$="{F5}"THEN PRINT"{RVSON}NICHT UE
    BERNOMMEN{RVOFF}":RETURN                              <163>
650 GOTO 580                                              <230>
660 RETURN                                                <210>
670 :                                                     <138>
680 REM *** DIRECTORY ***                                 <137>
690 PRINT"{CLR}";                                         <120>
700 CLOSE 2:OPEN 2,8,15:X=0:A$="":B$="":C=
    0:LE$=""                                              <096>
710 OPEN 1,8,0,"$0"                                       <096>
720 GET#1,A$,B$                                           <173>
730 GET#1,A$,B$                                           <183>
740 GET#1,A$,B$                                           <193>
750 C=0                                                   <151>
760 IF A$<>"" THEN C=ASC(A$)                              <120>
770 IF B$<>"" THEN C=C+ASC(B$)*256                        <199>
780 IF X=0 THEN TB=2:GOTO 800                             <073>
790 TB=5                                                  <080>
800 PRINT MID$(STR$(C),2);TAB(TB);                        <135>
810 GET#1,B$:IF ST<>0 THEN 950                            <252>
820 IF B$<>CHR$(34)THEN GOTO 810                          <169>
830 IF X=0 THEN PRINT"{RVSON}";                           <193>
840 PRINT CHR$(34);:PRINT CHR$(34);:PRINT"
    {DEL}";                                               <208>
850 GET#1,B$:IF B$<>CHR$(34)AND X=0 THEN P
    RINT"{RVSON}"B$;:GOTO 850:GOTO 870                    <092>
860 IF B$<>CHR$(34)THEN PRINT B$;:GOTO 850                <142>
870 IF X=0 THEN PRINT"{RVSON}";                           <233>
880 PRINT CHR$(34);:PRINT CHR$(34);:PRINT"
    {DEL}";                                               <248>
890 GET#1,B$:IF B$=CHR$(32) THEN 890                      <022>
900 PRINT TAB(24);:C$=""                                  <232>
910 C$=C$+B$:GET#1,B$:IF B$<>""  THEN 910                 <164>
920 LE$=LEFT$(C$,6):IF X=0 THEN X=1:GOSUB
    980: PRINT"{4LEFT,RVSON}"LE$:GOTO 940                 <028>
930 PRINT LE$                                             <072>
940 IF ST=0 THEN 730                                      <081>
950 PRINT"{LEFT}BLOCKS FREE":CLOSE 1                      <212>
960 GET T$:IF T$=""THEN 960                               <235>
970 T$="":RETURN                                          <228>
980 LE=LEN(C$):LE=6-LE                                    <037>
990 FOR I=1 TO LE:ZUS$=ZUS$+" ":NEXT                      <071>
1000 LE$=ZU$+LE$:RETURN                                   <033>
1010 REM *** TITEL ***                                    <110>
1020 POKE 53280,6 :POKE 53281,14:PRINT CHR
     $(142)CHR$(147);                                     <020>
1030 PRINT                                                <116>
1040 PRINT"{BLACK,SPACE}@@@{5SPACE}@@@ @@@
     @@@@ @@@{4SPACE}@@ @@{4SPACE}@@                      <225>
1050 PRINT" @@@@{3SPACE}@@@@ @@@@@@@ @@@@{
     3SPACE}@@ @@{4SPACE}@@                               <164>
1060 PRINT" @@ @@ @@ @@ @@"SPC(6)"@@ @@{2S
     PACE}@@ @@{4SPACE}@@                                 <109>
1070 PRINT" @@{2SPACE}@@@{2SPACE}@@ @@"SPC
     (6)"@@{2SPACE}@@ @@ @@{4SPACE}@@                     <129>
1080 PRINT" @@{3SPACE}@{3SPACE}@@ @@"SPC(6
     )"@@{3SPACE}@@@@ @@{4SPACE}@@                        <019>
1090 PRINT" @@"SPC(7)"@@ @@@@{4SPACE}@@{4S
     PACE}@@@ @@{4SPACE}@@                                <037>
1100 PRINT" @@"SPC(7)"@@ @@"SPC(6)"@@{5SPA
     CE}@@ @@{4SPACE}@@                                   <252>
1110 PRINT" @@"SPC(7)"@@ @@"SPC(6)"@@{5SPA
     CE}@@ @@{4SPACE}@@                                   <006>
1120 PRINT" @@"SPC(7)"@@ @@@@@@@ @@{5SPACE
     }@@ @@@@@@@@                                         <236>
1130 PRINT" @@"SPC(7)"@@ @@@@@@@ @@{5SPACE
     }@@{2SPACE}@@@@@@                                    <226>
1140 PRINT:PRINT                                          <076>
1150 PRINT"{2SPACE}WW{5SPACE}WW{2SPACE}WWW
     W{2SPACE}WW{2SPACE}W WWWWW WWWW                      <238>
1160 PRINT"{2SPACE}WWW{3SPACE}WWW WW{2SPAC
     E}WW WW W{2SPACE}WW{4SPACE}WW{2SPACE}
     W                                                    <096>
```


Listing 1. »MENUE-MAKER« lädt Programme menügesteuert. Bitte mit dem Checksummer (Seite 159) eingeben.

```
1170 PRINT"{2SPACE}HH H H HH HH{2SPACE}HH
     HHH{3SPACE}HH{4SPACE}HH{2SPACE}H          <106>
1180 PRINT"{2SPACE}HH{2SPACE}H{2SPACE}HH H
     HHHHH HHH{3SPACE}HHHH{2SPACE}HHHH         <113>
1190 PRINT"{2SPACE}HH{5SPACE}HH HH{2SPACE}
     HH HH H{2SPACE}HH{4SPACE}HH H             <025>
1200 PRINT"{2SPACE}HH{5SPACE}HH HH{2SPACE}
     HH HH{2SPACE}H HHHHH HH{2SPACE}H          <101>
1210 PRINT                                     <040>
1220 PRINT"{7SPACE}WRITTEN BY SIMON AND ED
     Y"                                        <203>
1250 GET A$:IF A$=""THEN 1250                  <157>
1260 PRINT"{CLR}"                              <232>
1270 PRINT,"{2DOWN,WHITE,2SPACE}IHRE WAHL
     :"                                        <110>
1280 PRINT,"{2DOWN,2SPACE}1{2SPACE}DIRECTO
     RY EINLESEN"                              <226>
1290 PRINT,"{2DOWN,2SPACE}2{2SPACE}UEBERNE
     HMEN"                                     <018>
1300 PRINT,"{2DOWN,2SPACE}3{2SPACE}INFO"       <160>
1310 PRINT,"{2DOWN,2SPACE}4{2SPACE}MENU SP
     EICHERN"                                  <033>
1320 PRINT,"{2DOWN,2SPACE}5{2SPACE}ENDE"       <089>
1330 GET A$ :IF A$=""THEN 1330                 <046>
1340 IF A$="1"THEN GOSUB 690:GOTO 1260         <103>
1350 IF A$="2"THEN GOSUB 350:GOTO 1260         <154>
1360 IF A$="3"THEN GOSUB 50:GOTO 1260          <061>
1370 IF A$="4"THEN GOTO 1460                   <011>
1380 IF A$="5"THEN 1410                        <079>
1390 GOTO 1330                                 <204>
1400 REM *** ENDE ***                          <217>
1410 POKE 214,23:POKE 211,11:SYS 58640:PRI
     NT"{RVSON,BLUE}SIND SIE SICHER ?{UP}"     <087>
1420 GET T$:IF T$="N"THEN 1260                 <231>
1430 IF T$="J"THEN SYS 64738                   <098>
1440 GOTO 1420                                 <238>
1450 REM *** DATEI SPEICHERN ***               <022>
1460 CLOSE 15:OPEN 15,8,15:PRINT#15,"S0:MS
     D":CLOSE 15:CLOSE 2:OPEN 2,8,2,"MSD,S
     ,W"                                       <223>
1470 Z=0                                       <201>
1480 Z=Z+1:IF Z=145 THEN CLOSE 2:GOTO 1530     <001>
1490 IF NA$(Z)=""THEN CLOSE 2:GOTO 1530        <004>
1500 PRINT#2,NA$(Z);NU$(Z)                     <118>
1510 GOTO 1480                                 <245>
1520 REM *** MENU SPEICHERN ***                <247>
1530 CLOSE 15:OPEN 15,8,15:PRINT#15,"S0:ME
     NU":CLOSE 15:PRINT"{CLR,UP}";:PRINT"R
     UN1600"                                   <004>
1540 PRINT"{3DOWN}POKE43,PEEK(61)+1:POKE44
     ,PEEK(62)"                                <054>
1550 PRINT"{2DOWN}SAVE"CHR$(34)"MENU"CHR$(
     34)",8"                                   <233>
1560 PRINT"{4DOWN}POKE43,1:POKE44,8"           <021>
1570 PRINT"{2DOWN}GOTO1260"                    <136>
1580 POKE 631,19:POKE 632,13:POKE 633,13:P
     OKE 634,13:POKE 635,13:POKE 636,13        <214>
1590 POKE 637,13:POKE 198,7:END                <022>
1600 STOP                                      <142>
1610 POKE 53280,14:POKE 53281,6                <042>
1620 PRINT"{CLR,9DOWN,WHITE,2DOWN,11SPACE}
     ICH LADE DIE DATEI":Z=0:DIM NA$(144),
     NU$(144)                                  <174>
1630 CLOSE 2:OPEN 2,8,2,"MSD,S,R"              <101>
1640 Z=Z+1                                     <184>
1650 INPUT#2,NAA$                              <192>
1660 IF RIGHT$(NAA$,1)="8"THEN NA$(Z)=LEFT
     $(NAA$,LEN(NAA$)-1):NU$(Z)=",8"           <068>
1670 IF RIGHT$(NAA$,2)="81"THEN NA$(Z)=LEF
     T$(NAA$,LEN(NAA$)-2):NU$(Z)=",8,1         <134>
1680 IF ST=64 THEN CLOSE 2:GOTO 1700           <250>
1690 GOTO 1640                                 <074>
1700 PRINT"{CLR}";:Z=0                         <014>
1710 PRINT"{2DOWN,14SPACE}N{RVSON}MENU-MAK
     ER{RVOFF}N"                               <059>
1720 PRINT"{DOWN,15SPACE}BY SIMON & EDY"       <014>
1730 PRINT TAB(5)"{2DOWN}CURSOR UP{4SPACE}
     NAECHSTES ELEMENT"                        <120>
1740 PRINT TAB(5)"{DOWN}CURSOR DOWN{2SPACE
     }VORHERIGES ELEMENT"                      <180>
1750 PRINT TAB(5)"{DOWN}RETURN{7SPACE}WAEH
     LEN"                                      <127>
1760 PRINT"{HOME,14DOWN,4SPACE}U**********
     ******I"                                  <129>
1770 PRINT"{4SPACE}-{16SPACE}-"                <141>
1780 PRINT"{4SPACE}J****************K"         <032>
1790 POKE 214,15:POKE 211,5:SYS 58640:PRIN
     T NA$(1):Z=1                              <182>
1800 GET T$:IF T$=""THEN 1800                  <235>
1810 IF T$="{UP}"THEN Z=Z+1:X=1:GOSUB 1890
     :GOTO 1800                                <140>
1820 IF T$="{DOWN}"THEN Z=Z-1:X=2:GOSUB 18
     90:GOTO 1800                              <122>
1830 IF T$=CHR$(13)THEN GOTO 1850              <175>
1840 GOTO 1800                                 <130>
1850 PRINT"{CLR,BLUE}";:PRINT"LOAD"CHR$(34
     )NA$(Z)CHR$(34)NU$(Z)                     <176>
1860 PRINT"{4DOWN}POKE646,14:RUN"              <082>
1870 POKE 631,19:POKE 632,13:POKE 633,13:P
     OKE 198,3:NEW                             <124>
1880 REM *** UNTERPROG ***                     <115>
1890 IF Z=0 THEN Z=1:RETURN                    <039>
1900 IF NA$(Z)=""AND X=2 THEN Z=Z+1:RETURN     <118>
1910 IF Z=145 THEN Z=144:RETURN                <200>
1920 IF NA$(Z)=""AND X=1 THEN Z=Z-1:RETURN     <025>
1930 POKE 214,15:POKE 211,5:SYS 58640:PRIN
     T"{16SPACE}"                              <188>
1940 POKE 214,15:POKE 211,5:SYS 58640:PRIN
     T NA$(Z):RETURN                           <126>
```

Listing 1. »MENUE-MAKER« (Schluß)



Sie wollen mit 32 Sprites und vier Bildschirmbereichen gleichzeitig arbeiten? Nichts leichter als das. Mit Provic 64 können simultan Grafik, Text oder veränderte Zeichensätze dargestellt werden.

Die Autoren (Jürgen und Stefan Haas) sind C64-Spezialisten, die sich bereits 1983 diesen Computer angeschafft haben. Mit ihren Erfahrungen wollen sie Ihnen weiterhelfen:

Schon kurze Zeit nach dem Kauf des Computers stellte sich der Frust über die schlechte Dokumentation bei uns ein. So saßen wir oft stundenlang vor dem Bildschirm, der nur undefinierbare Zeichen zeigte, weil wir bei dem Versuch, die Geheimnisse des C64 zu enträtseln, in irgendeinen unbekannten Darstellungsmodus gerieten. Dabei entdeckten wir, daß der C64 nicht nur acht, sondern auch 16, 24, 32 oder noch mehr Sprites gleichzeitig zeigen kann. Zusätzlich ergab sich die Möglichkeit, mehrere Bildschirmmodi zu mischen.

Wir entschlossen uns, den Kunstgriff, der dies ermöglicht, anderen C64-Fans nicht vorzuenthalten. Also entwickelten wir ein Programm in Maschinensprache und dazu ein kleines Demonstrationsprogramm in Basic.

Zum Programm

Durch Aufruf der Initialisierungsroutine wird der Interruptmechanismus des C64 verändert. Nicht mehr der Timer der CIA 1, sondern der VIC 6567 löst jetzt den Interrupt aus, und zwar synchron zum Takt des Bildschirmsignals. Außerdem werden vier sogenannte Pseudo-VICs eingerichtet. Alle POKEs, von Spritebewegung über Bildschirmfarbe bis zur Grafikkonfiguration, müssen ab jetzt in

PROV

diese Pseudo-VICs erfolgen. Jeder dieser Pseudo-VICs ist für einen der vier Bildschirmbereiche zuständig:

Der Bildschirm wird in vier waagerechte Bereiche aufgeteilt, deren Grenzlinien fast beliebig nach oben oder unten verschoben werden können. Jeder einzelne Bereich kann acht Sprites darstellen und eine eigene Farb- und Grafikkonfiguration aufweisen. So können zum Beispiel Normalschrift, Hires-Grafik, Multicolor-Grafik und eventuell ein selbstdefinierter Zeichensatz gleichzeitig auf dem Bildschirm dargestellt werden.

Provic 64 läßt sich selbstverständlich wieder abschalten (bei Kassetten- oder Diskettenoperationen nötig).

Für Maschinensprachefreaks nun eine kurze Funktionsbeschreibung der Interruptroutine:

Bei Aufruf der Einschaltroutine (SYS 52544) wird der IRQ-Vektor auf die Hauptroutine des Provic 64 gestellt und der bisherige Interrupt durch den Timer A der CIA 1 verboten, während der Raster-IRQ des VIC 6567 erlaubt wird.

Sobald der Bildschirmstrahl die eingestellte Rasterzeile erreicht, wird ein Interrupt ausgelöst, und der Prozessor bearbeitet die Hauptroutine. In dieser wird zunächst anhand eines Zählers ($ CFFF) festgestellt, welcher Bildschirmbereich an der Reihe ist. Dann wird die Rasterzeile, die das Ende dieses Bildschirms kennzeichnet, eingestellt.

Kurzinfo: Provic 64

Programmart: Tool für Sprites
Laden: LOAD "LADER",8
Start: Nach dem Laden RUN eingeben
Besonderheiten: Der Lader erzeugt den Maschinensprachteil »Provic 64« und speichert ihn automatisch auf Diskette. Das Zusatzprogramm »Demo« zeigt einige der Fähigkeiten von »Provic 64«.
Programmautoren: Stefan und Jürgen Haas

Anschließend werden, falls ein entsprechendes Flag gesetzt ist, die Sprite- und andere Bildschirmparameter in den VIC 6567 übertragen. Nach dem Weiterzählen des IRQ-Zählers ($ CFFF) wird entweder der Interrupt beendet oder zur IRQ-Routine des Betriebssystems gesprungen (nach jedem vierten Interrupt). So werden in der Sekunde 200 Interrupts (vier pro Fernsehbild) ausgelöst und 50mal in der Sekunde (einmal pro Bild) die normale IRQ-Routine abgearbeitet. Dadurch zählt die interne Uhr TI in 50stel Sekunden, und TI$ wird unbrauchbar.

Beim Aufruf der Ausschaltroutine wird der Raster-IRQ des VIC 6567 unterbunden, der Interrupt des Timers A in CIA 1 erlaubt und der IRQ-Vektor auf die IRQ-Routine des Betriebssystems eingestellt.

Im Grunde ist jeder der vier Pseudo-VICs wie der echte VIC zu behandeln. Ausnahmen sind hier nur die Register 30 (Sprite-Sprite-Kollision) und 31 (Sprite-Hintergrund-Kollision), die sich auf den jeweils vorausgegangenen Bildschirmbereich beziehen. Die Register 19 und 20 (Lightpenkoordinaten) sowie 25 und 26 (IRQ-Flags und -Maske) werden nicht behandelt, da diese Funktionen nur direkt über den VIC 6567 sinnvoll gehandhabt werden können. Außerdem hat jeder Pseudo-VIC noch zusätzliche Register für zwei Flags (REG 47 und REG 57), acht Sprite-Pointer (REG 48 bis REG 55), Videomatrix-Anfangsadresse High-Byte (REG 56) und die CIA 2, REG 0, Bits 0 und 1 (REG 58) (Adreßbits 14 und 15 des VIC 6567).

Die vier Basisadressen der PVICs sind:

PVIC 1	52992 ($ CF00)	= REG 0
PVIC 2	53056 ($ CF40)	= REG 0
PVIC 3	53120 ($ CF80)	= REG 0
PVIC 4	53184 ($ CFC0)	= REG 0

Da die Pseudo-VICs praktisch gleichberechtigt sind, hier die Registerbeschreibung eines Pseudo-VICs:

REG 0:	X-Koordinate des Sprite 0
REG 1:	Y-Koordinate des Sprite 0 Beachte: Die Y-Koordinaten sollten im Bereich des zugehörigen Bildschirmbereichs liegen, sonst ist der Sprite nicht zu sehen. Näheres siehe unten.
REG 2 bis 15:	Wie REG 0 und 1, aber für Sprites 1 bis 7
REG 16:	MSB (höchstes Bit) der X-Koordinaten
REG 17:	Bits 0 bis 2: Y-Abstand der Zeichen vom



Ein Demo der Fähigkeiten von Provic 64

I C 6 4

	oberen Bildrand in Rasterzeilen (Softscrolling!) Bit 3: Umschaltung 24/25-Zeilendarstellung Bit 4: Bild an/aus: Es hat keinen Sinn, das Bild teilweise ausschalten zu wollen, da der VIC dieses Bit nur einmal pro Fernsehbild prüft (also entweder das ganze Bild an oder aus) Bit 5: Hires-Grafikmodus an Bit 6: Hintergrundmehrfarb-Modus an Bit 7: Nummer der Interrupt-Rasterzeile Bit 8; es hat wenig Sinn, dieses Bit zu setzen, da so nur Rasterzeilen angesprochen werden, die unterhalb des Bildschirms liegen. Ist in irgendeinem Pseudo-VIC die 9-Bit-Zahl für die Rasterzeile größer als 311, so wird überhaupt kein IRQ mehr ausgelöst.
REG 18:	Nummer der Rasterzeile Bits 0 bis 7; hier ist anzugeben, wann der nächste Interrupt ausgelöst werden soll, das heißt wo der Bildschirmbereich dieses PVICs zu Ende sein soll. Dabei sollte folgende Reihenfolge eingehalten werden: REG 18: PVIC 1 < PVIC 2 < PVIC 3 <PVIC 4 (Zyklische Vertauschungen möglich!)
REG 19 und 20:	nicht verwendet (siehe oben)
REG 21:	Sprite enable (einschalten)

REG 22:	Bits 0 bis 2: Soltscrolling in X-Richtung Bit 3: Umschaltungen 38/40-Spaltendarstellung Bit 4: Multicolor-Modus ein Bit 5 bis 7: unbenutzt
REG 23:	Sprite vergrößern in Y-Richtung
REG 24:	Bits 1 bis 3: Adresse Zeichengenerator (Bits 11 bis 13) Bits 4 bis 7: Adresse Video-RAM (Bits 10 bis 13)
REG 25 und 26:	nicht verwendet (siehe oben)
REG 27:	Sprite-Priorität vor Hintergrund
REG 28:	Flags für Multicolor-Sprites
REG 29:	Sprite vergrößern in X-Richtung
REG 30:	Sprite-Sprite-Kollision
REG 31:	Sprite-Hintergrund-Kollision Achtung: geben die Kollisionen des vorangegangenen Bildschirmbereichs an: Findet im Bereich von PVIC 3 eine Kollision statt, wird dies im PVIC 4 registriert. Kollisionen im Bereich von PVIC 4 werden im PVIC 1 registriert. Dieses Register muß gelöscht werden, um neue Kollisionen anzeigen zu können!
REG 32:	Rahmenfarbe
REG 33 bis 36:	Hintergrundfarben 0 bis 3
REG 37 und 38:	Multicolor-Sprite-Farben 0 und 1

REG 39 bis 46:	Farben für Sprites 0 bis 7
REG 47:	Flag für Spritebehandlung; nur wenn der Inhalt dieses Registers nicht Null ist, werden die Register, die etwas mit Sprites zu tun haben, vom PVIC in den VIC 6567 übertragen. Das sind REG 0 bis REG 16, REG 21, REG 23, REG 27 bis REG 31, REG 37 bis 46 sowie REG 48 bis 55. Ist der Inhalt Null, gelten für die Sprites die Werte des vorherigen PVICs, während die Kollisionen erst im nächsten PVIC, in dem REG 48 ungleich Null ist, angezeigt werden.
REG 48 bis 55:	Sprite-Pointer für Sprites 0 bis 7; die Pointer auf die Bitmuster der Sprites werden nicht mehr in die Speicherzellen 2040 bis 2047 geschrieben, sondern in diese Register des PVICs.
REG 56:	In diesem Register muß das High-Byte der Video-RAM-Anfangsadresse plus 3 stehen; normalerweise also 4 + 3 = 7 (da der Bildschirm nach dem Einschalten des Computers bei 1024 beginnt, 1024 = $ 0400). Bei Verlegung des Video-RAMs ist also der Inhalt dieses Registers zu korrigieren.
REG 57:	Flag für Bildschirmparameter-Behandlung; nur wenn der Inhalt dieses Registers nicht Null ist, werden die REG 17, 22, 24 sowie 32 bis 36 und REG 58 in den VIC 6567 übertragen.
REG 58:	Bits 0 und 1: Adreßbits 14 und 15 des VIC 6557; werden nach CIA 2 REG 0 Bits 0 und 1 übertragen. Mit diesen Bits kann Video-RAM, Charaktergenerator, Grafik-Bitmap in 16-KByte-Schritten verschoben werden. Da die Bits low-aktiv sind, sind sie beim Einschalten gesetzt (also REG 58 = 3). Bits 2 bis 7: unbenutzt, immer 0.

Übergang eines Sprites zwischen zwei Bildschirmbereichen:

Soll ein Sprite zwischen zwei Bildschirmbereichen wechseln, muß in beiden Bereichen derselbe Sprite (also z. B. beidesmal Sprite 4) die gleiche Position besitzen, und zwar so lange, wie der Sprite die Trennlinie zwischen den Bereichen überdeckt. Beachtet man dies nicht, werden die entsprechenden Sprites zerschnitten und verschoben.

Aktivieren von Provic 64: Von Basic aus mit SYS 52544 und von Maschinensprache aus mit JSR $CD40.

Ausschalten von Provic 64: Von Basic aus mit SYS 52970 und von Maschinensprache aus mit JSR $CEEA.

Der Basic-Lader:

Der Lader erzeugt Provic 64 aus den DATA-Zeilen, und falls kein Prüfungsfehler vorliegt, wird Provic 64 sofort als Maschinenprogramm auf Floppy oder Datasette (Zeile 400 entsprechend ändern!) gespeichert. Dieses Maschinenprogramm enthält auch gleich die Standardwerte der Pseudo-VICs.

Das Demo-Programm:

Zunächst muß Provic 64 geladen werden. Dies erfolgt in Ihren eigenen Demo-Programmen am besten mit der Zeile »IF PEEK (52544) > < 120 THEN LOAD "PROVIC64", Gerätenummer, 1« die am Anfang des Basic-Programms stehen sollte.

Das Demonstrationsprogramm zeigt einige der Vorzüge von Provic 64. Es ist nur als Anregung gedacht, deshalb verzichten wir hier auf eine nähere Beschreibung.

Provic 64 ist nicht nur für Basic-Programmierer, sondern vor allem auch für Maschinensprache-Freaks gedacht, da erst durch schnelle Maschinenprogramme die Möglichkeiten von Provic 64 voll ausgeschöpft werden können.

(Jürgen und Stefan Haas/kn)

Tabellarische Übersicht zu Provic

Belegter Adreßraum:

$CD40	Einschaltroutine
$CD58	Interruptroutine
$CEEA	Ausschaltroutine
$CF00	Pseudo-VIC 1
$CF40	Pseudo-VIC 2
$CF80	Pseudo-VIC 3
$CFC0	Pseudo-VIC 4
$CFFF	Interruptzähler

Provic 64 einschalten:
in Basic: SYS 52544
in Maschinensprache: JSR $CD40
Provic 64 ausschalten:
in Basic: SYS 52970
in Maschinensprache: JSR $CEEA
Benutzte RAM-Adressen:
in der Zero-Page: 187 ($BB)
188 ($BC)
Rechenzeitzuwachs bei aktiviertem Provic 64:

alle Spriteflags (REG 47) und Bildschirmparameterflags (REG 57) gelöscht:	2,5 %
für jedes gesetzte Spriteflag (REG 47):	zirka +2,4 %
für jedes gesetzte Bildschirmflag (REG 57):	zirka +0,5 %
alle Sprite- und Bildschirmflags gesetzt:	zirka +15,0 %

Falls der Rechner abstürzt, rettet <Run-Stop/Restore>!
Die Zeitvariable TI zählt bei aktiviertem Provic 64 in 50stel Sekunden (statt 60stel);
TI$ wird somit unbrauchbar.
Zeiger für Interruptaussprung von PVIC 3 bis PVIC 4: $CEE5
Zeiger für Interruptaussprung von PVIC 1: $CEE8

```
10 REM INTERRUPT-ROUTINE ZUR ERZEUGUNG       <217>
11 REM                                       <073>
12 REM EINES MAXIMAL VIERTEILIGEN            <123>
13 REM                                       <075>
14 REM BILDSCHIRMES UND 32 SPRITES           <142>
15 REM                                       <077>
16 REM AUF EINEM COMMODORE 64 COMPUTER       <095>
17 REM                                       <079>
18 REM          BY GEBR. HAAS                <168>
19 REM                                       <081>
100 REM PRUEFSUMMEN-KONTROLLE                <016>
101 REM                                      <163>
110 RESTORE:PS=0                             <164>
120 FOR A=0 TO 511                           <015>
130 READ WERT                                <013>
140 PS=PS+WERT                               <145>
150 NEXT A                                   <170>
160 IF PS<>60913 THEN PRINT"DATA-PRUEFSUMM
    E FALSCH !":END:*                        <197>
199 REM                                      <005>
200 REM MASCHINENCODE UEBERTRAGEN            <029>
201 REM                                      <007>
210 RESTORE                                  <004>
220 FOR A=0 TO 447                           <002>
230 READ WERT                                <113>
240 POKE 52544+A,WERT                        <048>
250 NEXT A                                   <014>
260 FOR A=0 TO 63                            <190>
265 READ WERT                                <150>
270 FOR B=0 TO 3                             <027>
280 POKE 52992+B*64+A,WERT                   <045>
290 POKE 52992+B*64+18,95+50*B               <217>
295 NEXT B,A                                 <047>
299 REM                                      <107>
300 REM PROVIC 64 ABSPEICHERN                <010>
301 REM                                      <109>
```

Listing 1. »LADER« erzeugt automatisch den Maschinensprachteil »Provic 64« und speichert ihn auf Diskette

```
310 N$="PROVIC 64"                             <233>
320 FOR A=1 TO LEN(N$)                          <074>
330 POKE 49151+A,ASC(MID$(N$,A,1))              <107>
340 NEXT A                                      <106>
350 POKE 183,LEN(N$)                            <185>
360 POKE 187,0:POKE 188,192                     <237>
370 POKE 185,1                                  <039>
380 POKE 193,64:POKE 194,205                    <234>
390 POKE 174,0:POKE 175,208                     <037>
400 POKE 186,8: REM FUER FLOPPY    : 8
                    FUER DATASETTE: 1           <021>
410 SYS 62957:  REM SAVE ROUTINE                <217>
420 END                                         <168>
999 REM                                         <043>
1000 REM PROVIC 64 MASCHINENCODE                <140>
1001 REM                                        <045>
1002 DATA 120,169,88,162,205,141,20,3,142,
     21,3,169,1,141,13,220,141,26,208           <082>
1003 DATA 141,255,207,88,96,169,1,141,25,2
     08,172,255,207,240,15,136,240,9,136        <057>
1004 DATA 240,3,162,192,44,162,128,44,162,
     64,44,162,0,189,18,207,141,18,208          <038>
1005 DATA 189,47,207,208,3,76,141,206,173,
     30,208,29,30,207,157,30,207,173,31         <234>
1006 DATA 208,29,31,207,157,31,207,189,21,
     207,141,21,208,189,23,207,141,23           <000>
1007 DATA 208,189,29,207,141,29,208,189,0,
     207,141,0,208,189,1,207,141,1,208          <198>
1008 DATA 189,2,207,141,2,208,189,3,207,14
     1,3,208,189,4,207,141,4,208,189,5          <007>
1009 DATA 207,141,5,208,189,6,207,141,6,20
     8,189,7,207,141,7,208,189,8,207,141        <039>
1010 DATA 8,208,189,9,207,141,9,208,189,10
     ,207,141,10,208,189,11,207,141,11          <167>
1011 DATA 208,189,12,207,141,12,208,189,13
     ,207,141,13,208,189,14,207,141,14          <113>
1012 DATA 208,189,15,207,141,15,208,189,16
     ,207,141,16,208,189,27,207,141,27          <047>
1013 DATA 208,189,28,207,141,28,208,189,37
     ,207,141,37,208,189,38,207,141,38          <255>
1014 DATA 208,189,39,207,141,39,208,189,40
     ,207,141,40,208,189,41,207,141,41          <052>
1015 DATA 208,189,42,207,141,42,208,189,43
     ,207,141,43,208,189,44,207,141,44          <018>
1016 DATA 208,189,45,207,141,45,208,189,46
     ,207,141,46,208,189,56,207,133,188         <202>
1017 DATA 169,248,133,187,160,0,189,48,207
     ,145,187,200,189,49,207,145,187,200        <038>
1018 DATA 189,50,207,145,187,200,189,51,20
     7,145,187,200,189,52,207,145,187           <065>
1019 DATA 200,189,53,207,145,187,200,189,5
     4,207,145,187,200,189,55,207,145           <155>
1020 DATA 187,189,57,207,240,67,189,58,207
     ,41,3,157,58,207,173,0,221,41,252          <239>
1021 DATA 29,58,207,141,0,221,189,17,207,1
     41,17,208,189,22,207,141,22,208,189        <154>
1022 DATA 24,207,141,24,208,189,32,207,141
     ,32,208,189,33,207,141,33,208,189          <180>
1023 DATA 34,207,141,34,208,189,35,207,141
     ,35,208,189,36,207,141,36,208,224          <070>
1024 DATA 192,208,5,169,255,141,255,207,23
     8,255,207,138,240,3,76,129,234,76          <201>
1025 DATA 49,234,120,169,49,162,234,141,20
     ,3,142,21,3,169,129,141,13,220,169         <006>
1026 DATA 0,76,80,205,234,0,0,0,0,0,0,0,0,
     0,0,0,0,0,0,0,0,0,27,95,0,0,0,0,200        <113>
1027 DATA 0,21,121,240,0,0,0,0,0,14,6,1,2,
     3,4,0,1,2,3,4,5,6,7,8,0,0,0,0,0,0          <127>
1028 DATA 0,0,0,7,1,3,0,0,0,0,0                 <076>
```

Listing 1. (Schluß)

```
6 REM  DIESES KURZE DEMO-PROGRAMM SOLL          <075>
8 REM  EIN PAAR DARSTELLUNGSFORMEN              <251>
10 REM ZEIGEN, WIE SIE MIT PROVIC-64            <024>
12 REM RELATIV EINFACH ERREICHBAR SIND.         <110>
14 REM          BY GEBR. HAAS                   <164>
19 REM                                          <081>
20 REM LADEN DER PROVIC-64 ROUTINE              <069>
21 REM                                          <083>
30 IF PEEK(52544)=120 THEN 100                  <014>
40 PRINT"{2DOWN,SPACE}AUF WELCHEM DATENTRA
   EGER IST PROVIC-64"                          <056>
50 PRINT"{DOWN,SPACE}VERFUEGBAR ( FLOPPY =
   8 / DATASETTE =1 )"                          <241>
60 INPUT" ";A$:A=VAL(A$)                        <073>
70 IF A<>1 AND A<>8 THEN 40                     <077>
80 LOAD"PROVIC 64",A,1                          <130>
99 REM                                          <161>
100 REM PSEUDO-VIC'S INITIALISIEREN             <145>
101 REM                                         <163>
110 P1=52992:P2=53056:P3=53120:P4=53184:RE
    M BASISADRESSEN DER PSEUDO-VIC'S            <206>
120 POKE P1+21,255:POKE P1+24,22:POKE P1+4
    7,1:POKE P1+27,255:POKE P2+21,255           <172>
130 POKE P2+17,59:POKE P2+24,24:POKE P2+32
    ,7:POKE P2+47,1:POKE P2+27,255              <071>
140 POKE P3+21,255:POKE P3+32,9:POKE P3+47
    ,1:POKE P3+27,255:POKE P4+32,5              <132>
150 POKE P4+21,255:POKE P4+24,22:POKE P4+4
    7,1:POKE P4+18,230:POKE P4+27,255           <217>
199 REM                                         <005>
200 REM SPRITE-DATEN UEBERTRAGEN                <034>
201 REM                                         <007>
210 RESTORE                                     <004>
220 FOR A=0 TO 126                              <124>
230 READ WERT                                   <113>
240 POKE 832+A,WERT                             <152>
250 NEXT A                                      <014>
299 REM                                         <107>
300 REM BILSCHIRM AUFBAUEN                      <127>
301 REM                                         <109>
310 PRINT"{CLR,WHITE,DOWN,RVSON,7RIGHT}P{S
    HIFT-SPACE}R{SHIFT-SPACE}O{SHIFT-SPACE
    }V{SHIFT-SPACE}I{SHIFT-SPACE}C{3SHIFT-
    SPACE}6{SHIFT-SPACE}4{3SHIFT-SPACE}D{S
    HIFT-SPACE}E{SHIFT-SPACE}M{SHIFT-SPACE
    }O"                                         <192>
320 PRINT"{2DOWN,SPACE}IN DIESEM BEREICH:
    GROSS/KLEIN-SCHRIFT{DOWN}":                 <044>
330 FOR A=0 TO 7:PRINT"{BLUE}RRRRRRRRRRRR
    RRRRRRRRRRRRRRRRRRRRRRRRR{WHITE}";:N
    EXT A                                       <178>
335 FOR A=0 TO 14:POKE 1289+A,34:NEXT A         <248>
340 PRINT"{6SPACE}↑ DORT OBEN: HIRES-GRAFI
    K-MODUS ↑"                                  <024>
350 PRINT"{DOWN,13SPACE}HIER: GROSS-SCHRIF
    T/GRAFIK"                                   <093>
360 PRINT"{DOWN,SPACE}QWERTYUIO@*↑LKJHFSAZ
    XCVBNMQWERTFGZXCVB"                         <037>
370 PRINT"{5DOWN,6SPACE}↑ DORT OBEN: LAUFS
    CHRIFT ↑{3UP}":PRINT TAB(6);                <055>
390 SYS 52544:REM PROVIC-64 ACTIVIEREN          <080>
399 REM                                         <207>
400 REM HIRES-GRAFIK ZEICHNEN                   <201>
401 REM                                         <209>
410 FOR A=10110 TO 12360:POKE A,0:NEXT A        <212>
420 FOR A=0 TO 8*↑STEP.08                       <248>
430 X=3+A/.08:Y=77-11*SIN(A)-9*COS(A/.7)        <121>
440 AV=8192+320*INT(Y/8)+(Y AND 7)+8*INT(X
    /8)                                         <172>
450 POKE AV,PEEK(AV) OR 2↑(7-(X AND 7))         <072>
460 NEXT A                                      <226>
470 LA$="*** VON HAASISOFT ***{2SPACE}FUER
     DAS 64'ER MAGAZIN{3SPACE}"                 <128>
480 LA$=LA$+"***{3SPACE}P{SHIFT-SPACE}R{SH
    IFT-SPACE}O{SHIFT-SPACE}V{SHIFT-SPACE}
    I{SHIFT-SPACE}C{3SHIFT-SPACE}6 4{3SPAC
    E}"                                         <187>
490 LA$=LA$+LEFT$(LA$,25):R=53266               <066>
499 REM                                         <051>
500 REM DEMONSTRATINS-SCHLEIFE                  <141>
501 REM                                         <053>
510 REM SPRITES SETZEN                          <013>
511 REM                                         <063>
520 FOR A=0 TO 7                                <017>
530 POKE P1+2*A,30+24*A+7*RND(1):POKE P1+2
    *A+1,60+6*RND(1)                            <158>
540 POKE P1+39+A,RND(1)*16:POKE P1+48+A,13
    .5+RND(1)                                   <110>
```

Listing 2. Einige der Vorzüge von »Provic 64« zeigt das Programm »Demo«

```
550 POKE P2+2*A,30+24*A+7*RND(1):POKE P2+2
    *A+1,110+6*RND(1)                           <011>
560 POKE P2+39+A,RND(1)*16:POKE P2+48+A,13
    .5+RND(1)                                   <178>
570 POKE P3+2*A,30+24*A+7*RND(1):POKE P3+2
    *A+1,160+6*RND(1)                           <135>
580 POKE P3+39+A,RND(1)*16:POKE P3+48+A,13
    .5+RND(1)                                   <246>
590 POKE P4+2*A,30+24*A+7*RND(1):POKE P4+2
    *A+1,207+6*RND(1)                           <059>
600 POKE P4+39+A,RND(1)*16:POKE P4+48+A,13
    .5+RND(1)                                   <058>
610 NEXT A                                      <122>
619 REM                                         <173>
620 REM LAUFSCHRIFT SETZEN                      <097>
621 REM                                         <175>
625 FOR LP=1 TO LEN(LA$)-25                     <195>
630 LZ=LZ-1:IF LZ>0 THEN POKE P4+22,LZ OR
    8:FOR A=0 TO 9:NEXT A:GOTO 630              <110>
640 PRINT TAB(6);:WAIT 53265,128:WAIT 5326
    6,64:POKE 53206,15:PRINT LF$:PRINT"{2U
    P}"                                         <193>
660 LZ=7:LF$=MID$(LA$,LP,25)                    <244>
670 NEXT LP                                     <019>
680 GET A$:IF A$=""THEN 500                     <173>
690 SYS 52970:REM PROVIC-64 DESAKTIVIEREN       <194>
999 REM                                         <043>
1000 REM SPRITE-DATEN                           <203>
1001 REM                                        <045>
1002 DATA 0,0,0,0,126,0,1,255,128,7,255,22
     4,15,255,240,15,253,240,31,255,248         <077>
1003 DATA 31,255,248,63,255,252,63,255,252
     ,63,243,252,63,252,0,63,255,252,63         <031>
1004 DATA 255,252,31,255,248,31,255,248,15
     ,255,240,15,255,240,7,255,224,1,255        <199>
1005 DATA 128,0,126,0,0,0,0,0,126,0,1,25
     5,128,7,255,224,15,255,240,15,251          <154>
1006 DATA 240,31,255,248,31,255,248,63,255
     ,240,63,255,0,63,240,0,63,252,0,63         <003>
1007 DATA 255,0,63,255,224,31,255,248,31,2
     55,248,15,255,240,15,255,240,7,255         <050>
1008 DATA 224,1,255,128,0,126,0                 <079>
```

Listing 2. (Schluß)

MASKENGENERATOR FÜR VERWÖHNTE



Bild 1. Dieses Titelbild erscheint nach dem Starten des Maskengenerators

Ab jetzt ist das Gestalten von Bildschirmmasken ein Kinderspiel. Mit diesem Editorprogramm kreieren Sie schnell und komfortabel eine neue Bildschirmmaske und generieren die entsprechenden Basic-Zeilen.

Bei diesem Bildschirmmasken-Generator (Listing 1) kommt es nicht so auf Geschwindigkeit an, vielmehr sind Komfort und eine möglichst perfekt programmierte Maske die Schwerpunkte des Entwicklers. Der Maskengenerator bietet intelligente Optimierungsroutinen und erzeugt ein möglichst platzsparendes Basic-Programm für die generierte Bildschirmmaske. Dabei verhindert der Generator automatisch, daß die erzeugte Maske beim Erreichen des letzten Bildschirmzeichens um eine Zeile nach oben scrollt.

Das Listing besteht aus zwei unabhängigen Programmen. Das zweite Programm ist der schon erwähnte komfortable Editor. Er eignet sich hervorragend zum Zeichnen von Blockgrafik-Bildern. Es lassen sich Linien zeichnen, Zeilen löschen und zentrieren, Spalten löschen, Hintergrund- und Rahmenfarben umschalten, kurz: Er ist viel komfortabler als der Basic-Editor!

Programmbeschreibung

Der Maskengenerator wird mit »LOAD"Maske",« geladen und mit RUN gestartet. Es stehen folgende Kommandos zur Verfügung:

- alle in Basic erreichbaren Tasten außer CHR$(34) = » " «, denn ich glaube, daß Anführungszeichen in einer Maske nichts zu suchen haben

- F1	- Rahmenfarbe weiterschalten
- F3	- Hintergrundfarbe weiterschalten
- F5	- Wiederholung für alle Tasten ein
- F7	- Wiederholung für alle Tasten aus
- alle Cursor-Tasten	
- CBM+SHIFT	- Zeichensatz umschalten
- Cursor-Farben	- wie in Basic
- CTRL+Z	- Zeile zentrieren
- CTRL+X	- Zeile löschen
- CTRL+Y	- Spalte löschen
- CTRL+L	- Linie zeichnen:

Vor dem Erstellen einer neuen Maske sollte der Bildschirm mit <SHIFT CLR/HOME> gelöscht werden.

Wollen Sie eine horizontale oder vertikale Zeichenkette erzeugen, so schreiben Sie an die Anfangsposition das entsprechende Zeichen. Nun muß man mit dem Cursor wieder auf dieses Zeichen fahren und <CTRL L> (für Linie) drücken. Jetzt bewegen Sie den Cursor an das Ende der Zeichenkette und drücken <RETURN>. Die Zeichenkette wird erzeugt und der Cursor befindet sich an dessen Ende.

- REVERSE	- Tasten wie in Basic
- INSERT	- fügt an die Stelle, an der sich der Cursor befindet, ein SPACE ein und verschiebt die restliche Zeile nach rechts
- DELETE	- Cursor ein Zeichen nach links. Das Zeichen, auf dem sich der Cursor vorher befand, wird durch die Zeile, die nun eingerückt wird, gelöscht.
- CTRL+ ←	- Maske erzeugen.

Mit <CTRL ←> wird das Basic-Programm erzeugt. Damit Sie sehen, wie weit der Generator mit seiner Arbeit ist, wird das ausgelesene Zeichen durch ein weißes Sternchen überschrieben.

Nun folgen die Fragen nach Startzeile (Nummer der ersten Basic-Zeile) und Schrittweite, und die Maske wird gelistet. Hier können Sie die Maske ausprobieren und abspeichern.

Noch ein paar Hinweise: Am Anfang wird folgende Frage gestellt (Bild 1):

»Sollen 'SHIFT/SPACE' den 'SPACE'-Zeichen gleich sein?«

Falls Sie die SHIFT/SPACE-Taste nicht für irgendwelche Zwecke benötigen, können Sie einfach <RETURN> eingeben, sonst <N> + <RETURN>.

Beschreibung der Programmfunktion:

Speicheraufteilung:

Speicherzelle 2 und die Speicherzellen 832 bis 839 dienen als Flags und Zeiger zwischen den verschiedenen Programmteilen. Ihre Funktion:

2	= falls die Anfangsfrage mit ja beantwortet wird, wird sie auf 1 gesetzt
832	= letztes Zeichen
833	= Farbe des letzten Zeichens
834	= 1, wenn letztes Zeichen vorhanden
835	= Variablenstart der Maske Low-Byte
836	= Variablenstart der Maske High-Byte
837	= Zeichensatz (Wert aus 53272)
838	= Erste Farbe zur Generierung der ersten Zeile
839	= Falls letzte Zeile Zeichen enthält, dann gesetzt

$0800 - $8000	Bildschirmmaskengenerator
$8000 - $A000	Generierte Bildschirmmaske
$C000 - $D000	wird als Zwischenspeicher im PASS 1 benutzt

Der Generator arbeitet folgende Schritte ab:

Maske einlesen und vorbehandeln, Maske verarbeiten und als generiertes Programm nach $8000 schreiben. Basic-Start nach $8000.

Wenn Sie den genauen Ablauf studieren wollen, sind die REM-Zeilen eine wertvolle Hilfe.

Programmbehandlung

Um das Programm zu beschleunigen, sollte man es compilieren (auf der Programmservice-Diskette ist eine compilierte Version enthalten).

Wenn Sie eine Maske gespeichert haben, kann man folgendermaßen ins Hauptprogramm zurückkehren, ohne es neu laden zu müssen:

Compilierte Version: POKE 44,8:RUN

Basic-Version: SYS 64738, danach den Einzeiler für RENEW von H. und M. Sprave:

POKE2050,8:SYS42291:POKE46,PEEK(35)-(PEEK(781)
>253):POKE45,PEEK(781)
+2AND255:CLR:RUN

(Roman Barthe/kn)

Kurzinfo: Maske

Programmart: Programmier-Tool
Laden mit: LOAD "MASKE",8
Starten mit: Nach dem Laden RUN eingeben
Eingaben über: Tastatur
Besonderheiten: Komfortables Generieren von Basic-Routinen für Bildschirmmasken
Programmautor: Roman Barthe

```
10 REM  ******************************     <148>
15 REM  *  BILDSCHIRMMASKENGENERATOR  *     <239>
20 REM  *      MIT INTELLIGENTEN      *     <069>
25 REM  *  OPTIMIERUNGSEIGENSCHAFTEN  *     <078>
30 REM  *    UND GROESSTMOEGLICHER    *     <006>
35 REM  *   SPEICHERPLATZ-ERSPARNIS   *     <249>
40 REM  *         VERSION 2.0         *     <036>
45 REM  * WRITTEN 3/86 BY ROMAN BARTKE*     <242>
50 REM  *     MAX-VON-LAUE-STR.29     *     <035>
55 REM  *      5630 REMSCHEID 11      *     <091>
60 REM  *      TEL. 02191/63753       *     <223>
65 REM  *                             *     <114>
70 REM  *Z.150 EINZ. V.K.SMOCZYK 12/85*     <041>
75 REM  ******************************     <213>
80 :                                        <056>
85 :                                        <061>
90 POKE 55,255:POKE 56,127:CLR:REM BASICRA
   M-ENDE NACH 32768                        <076>
95 GOTO 755                                 <055>
100 REM PASS 1                              <014>
105 POKE 837,PEEK(53272):POKE 53281,G:POKE
     53280,BO                               <211>
110 CLR:FOR I=704 TO 704+63:READ Q:NEXT     <172>
115 DIM C$(15):FOR I=0 TO 15:READ C$:C$(I)
    =C$:NEXT                                <121>
120 ZA=24:SA=39:R=0:F=16:S=0:M=49155        <129>
125 IF PEEK(2023)=32 OR(PEEK(2023)=96 AND
    PEEK(2)=1)THEN POKE 834,0:GOTO 135      <148>
130 POKE 832,PEEK(2023):POKE 833,PEEK(5629
    5)AND 15:POKE 834,1                     <031>
135 FOR Z=0 TO ZA:B=0:FOR S=0 TO SA         <222>
140 X=PEEK(40*Z+S+1024):C=PEEK(40*Z+S+5529
    6)AND 15                                <138>
145 POKE 55296+40*Z+S,1:POKE 1024+40*Z+S,A
    SC("*")                                 <061>
150 XX=X+(X>127)*128:W=XX-(XX>-1 AND XX<32
     OR XX>95)*64-(XX>63 AND XX<96)*32      <216>
155 IF PEEK(2)=1 AND W=160 THEN W=32        <157>
160 IF R=0 AND X>127 THEN R=1:M=M+1:POKE M
    ,ASC("{RVSON}"):B=1:GOTO 170            <242>
165 IF R=1 AND X<127 THEN R=0:M=M+1:POKE M
    ,ASC("{RVOFF}")                         <173>
170 IF C=F THEN 190                         <255>
175 IF(W=32 OR W=160)AND R=0 THEN 190       <003>
180 IF EF=0 THEN POKE 838,ASC(C$(C)):EF=1:
    F=C:GOTO 190:REM ERSTE FARBE            <120>
185 F=C:M=M+1:POKE M,ASC(C$(C))             <119>
190 M=M+1:POKE M,W                          <181>
195 IF W<>32 OR R=1 THEN B=1:REM FALLS LEE
    RZEILE,DANN FLAG FUER CRSR-DOWN         <168>
200 NEXT                                    <210>
205 IF B=1 THEN 215                         <251>
210 M=M-39:POKE M,ASC("{DOWN}"):GOTO 230    <010>
215 M=M+1                                   <022>
220 IF(W=32 OR(W=96 AND PEEK(2)=1))AND R=0
     THEN POKE M,13:GOTO 230                <192>
225 POKE M,141                              <054>
230 IF Z=24 AND B=1 THEN POKE 839,1         <089>
235 NEXT                                    <245>
240 POKE 49152,INT(M/256):POKE 49153,M-PEE
    K(49152)*256                            <226>
245 REM PASS 2                              <032>
250 CLR                                     <108>
255 DIM Z$(51)                              <064>
260 M=PEEK(49152)*256+PEEK(49153)           <006>
265 FOR I=49156 TO M                        <143>
270 IF CHR$(PEEK(I))="{RVSON}"THEN R=1      <229>
275 IF CHR$(PEEK(I))="{RVOFF}"THEN R=0      <240>
280 IF PEEK(I)=141 THEN 295                 <221>
285 IF PEEK(I)=13 THEN GOSUB 375:GOTO 295   <079>
290 Z$=Z$+CHR$(PEEK(I)):GOTO 365            <117>
295 REM SPC(X) EINBAUEN (WENN MEHR ALS 7 S
    PACES)                                  <106>
300 L=0:S=0                                 <155>
305 L=L+1:IF MID$(Z$,L,1)=CHR$(32)AND R=0
    THEN S=S+1:GOTO 305                     <045>
310 IF S<8 THEN 320                         <086>
315 Z$=CHR$(153)+CHR$(166)+RIGHT$(STR$(S),
    LEN(STR$(S))-1)+")"+CHR$(34)+RIGHT$(Z$
    ,LEN(Z$)-S)                             <233>
320 IF LEN(Z$)<=65 THEN 345                 <024>
325 J=J+1:Z$(J)=LEFT$(Z$,65)+CHR$(34)+";"   <165>
330 J=J+1:Z$(J)=RIGHT$(Z$,LEN(Z$)-65)       <124>
335 IF PEEK(I)=141 THEN Z$(J)=Z$(J)+CHR$(3
    4)+";"                                  <009>
340 GOTO 360                                <126>
345 J=J+1                                   <094>
350 IF H=1 THEN H=0:Z$(J)=Z$:GOTO 360       <209>
355 IF H=0 THEN Z$(J)=Z$+CHR$(34)+";"       <136>
360 Z$="":Z=0                               <135>
365 NEXT:GOTO 410                           <204>
370 REM HINTERE SPACES BESEITIGEN           <096>
375 IF R=1 OR RIGHT$(Z$,1)<>" "THEN RETURN  <219>
380 D=LEN(Z$)                               <054>
385 D=D-1                                   <046>
390 IF MID$(Z$,D,1)=CHR$(32)THEN 385        <075>
395 Z$=LEFT$(Z$,D)                          <104>
400 H=1:RETURN                              <069>
405 :                                       <127>
410 REM FOR-NEXT EINBAUEN (AB 3 GLEICHEN Z
    EILEN)                                  <025>
415 DIM ZZ$(55):E=1:Z=1:REM 55=J+4 MAXIMAL
    E ZEILENANZAHL                          <178>
420 A=E:E=A+1:IF E>J+1 THEN 510             <013>
425 IF Z$(A)=Z$(E)THEN E=E+1:GOTO 425       <175>
430 REM A,E-A,Z$(A) STEHEN HIER BEREIT      <021>
435 IF LEFT$(Z$(A),2)=CHR$(153)+CHR$(166)T
    HEN SP=1:GOTO 445                       <223>
440 SP=0                                    <186>
445 IF(E-A)<>1 THEN 460                     <007>
450 Z=Z+1:IF SP=0 THEN ZZ$(Z)=CHR$(153)+CH
    R$(34)                                  <245>
455 ZZ$(Z)=ZZ$(Z)+Z$(A):GOTO 420            <004>
460 IF(E-A)<>2 THEN 485                     <156>
465 Z=Z+2:IF SP=0 THEN ZZ$(Z-1)=CHR$(153)+
    CHR$(34)                                <088>
470 ZZ$(Z-1)=ZZ$(Z-1)+Z$(A)                 <070>
475 IF SP=0 THEN ZZ$(Z)=CHR$(153)+CHR$(34)  <057>
480 ZZ$(Z)=ZZ$(Z)+Z$(A+1):GOTO 420          <167>
485 Z=Z+3:ZZ$(Z-2)=CHR$(129)+"I"+CHR$(178)
    +"1"+CHR$(164)                          <235>
490 ZZ$(Z-2)=ZZ$(Z-2)+RIGHT$(STR$(E-A),LEN
    (STR$(E-A))-1)                          <163>
495 IF SP=0 THEN ZZ$(Z-1)=CHR$(153)+CHR$(3
    4)                                      <164>
500 ZZ$(Z-1)=ZZ$(Z-1)+Z$(A):ZZ$(Z)=CHR$(13
    0)                                      <113>
505 GOTO 420                                <235>
510 REM ERSTE ZEILE GENERIEREN              <101>
515 EZ$=CHR$(151)+"53281,"+RIGHT$(STR$(PEE
    K(49154)),LEN(STR$(PEEK(49154)))-1)     <213>
520 EZ$=EZ$+":"+CHR$(151)+"53280,"          <184>
525 EZ$=EZ$+RIGHT$(STR$(PEEK(49155)),LEN(S
    TR$(PEEK(49155)))-1)                    <074>
530 EZ$=EZ$+":"+CHR$(151)+"53272,"+RIGHT$(
    STR$(PEEK(837)),LEN(STR$(PEEK(837)))-1
    )                                       <088>
535 EZ$=EZ$+":"+CHR$(153)+CHR$(34)+"{CLR}"
    +CHR$(PEEK(838))+CHR$(34)+";"           <224>
540 ZZ$(1)=EZ$:EZ$=""                       <185>
545 REM GENERIERUNG DER LETZTEN ZEILE       <128>
550 IF PEEK(839)=1 AND PEEK(834)=0 AND ZZ$
    (Z)<>CHR$(130)THEN ZZ$(Z)=ZZ$(Z)+"{HOM
    E}":GOTO 650                            <216>
555 REM SCHLEIFE GEHT NICHT BIS ZUR LETZTE
    N ZEILE                                 <254>
560 IF ZZ$(Z)=CHR$(130)AND PEEK(834)=0 THE
    N GOSUB 590:ZZ$(Z)=ZZ$(Z)+"{HOME}":GOT
    O 650                                   <183>
565 REM LETZTES ZEICHEN POKEN, FALLS DAS Z
    EICHEN NICHT SPACE IST                  <157>
570 IF PEEK(834)=0 THEN 650                 <145>
```

Listing 1. »Maske«, der komfortable Maskengenerator mit eingebauten Optimierungseigenschaften

```
575 IF ZZ$(Z)<>CHR$(130)THEN 625                <247>
580 GOSUB 590:GOTO 625                          <128>
585 REM FOR-NEXT SCHLEIFE UM 1 ERNIEDRIGEN      <109>
590 Z$=ZZ$(Z-2):IF VAL(RIGHT$(Z$,2))=0 THE
    N AN=1:GOTO 600                             <027>
595 AN=2                                        <063>
600 Z1$=LEFT$(Z$,LEN(Z$)-AN)                    <181>
605 Z$=STR$(VAL(RIGHT$(Z$,AN))-1):ZA$=RIGH
    T$(Z$,LEN(Z$)-1):ZZ$(Z-2)=Z1$+ZA$           <131>
610 IF AN=1 AND ZA$="2"THEN ZZ$(Z-2)=ZZ$(Z
    -1):ZZ$(Z)=ZZ$(Z-1):GOTO 620                <060>
615 Z=Z+1:ZZ$(Z)=ZZ$(Z-2)                       <172>
620 RETURN                                      <170>
625 ZZ$(Z)=LEFT$(ZZ$(Z),LEN(ZZ$(Z))-3)+"{H
    OME}"+RIGHT$(ZZ$(Z),2)                      <193>
630 Z=Z+1:ZZ$(Z)=CHR$(151)+"2023,"+RIGHT$(
    STR$(PEEK(832)),LEN(STR$(PEEK(832)))-1
    )                                           <160>
635 ZZ$(Z)=ZZ$(Z)+":"+CHR$(151)+"56295,"        <245>
640 ZZ$(Z)=ZZ$(Z)+RIGHT$(STR$(PEEK(833)),L
    EN(STR$(PEEK(833)))-1)                      <083>
645 REM MASKE ALS PROGRAMM                      <227>
650 PRINT"{CLR,GREY 3}";:POKE 53281,11:POK
    E 53280,11:POKE 53272,21                    <050>
655 PRINT"{HOME,3DOWN,3RIGHT}STARTZEILE{3S
    PACE}?"                                     <231>
660 PRINT"{3RIGHT,DOWN}SCHRITTWEITE ?{LEFT
    ,2UP}";:INPUT SZ$:PRINT"{DOWN}"SPC(16)
    ;:INPUT SW$                                 <109>
665 SZ=VAL(SZ$):SW=VAL(SW$)                     <244>
670 IF(SZ+Z>63999)OR(Z*SW+SZ-1>63999)OR SW
    <1 THEN 655                                 <198>
675 EZ=SZ+Z*SW-1:V=0:KA=32768                   <135>
680 FOR ZN=SZ TO EZ STEP SW:V=V+1               <012>
685 K=KA+LEN(ZZ$(V))+6                          <128>
690 KA=KA+1:POKE KA,K-INT(K/256)*256:KA=KA
    +1:POKE KA,INT(K/256)                       <223>
695 KA=KA+1:POKE KA,ZN-INT(ZN/256)*256:KA=
    KA+1:POKE KA,INT(ZN/256)                    <095>
700 FOR I=1 TO LEN(ZZ$(V)):KA=KA+1:POKE KA
    ,ASC(MID$(ZZ$(V),I,1)):NEXT                 <101>
705 KA=KA+1:POKE KA,0:NEXT                      <214>
710 KA=KA+2:POKE KA-1,0:POKE KA,0               <132>
715 REM VON 2049 AUF 32769 UMSCHALTEN           <151>
720 KA=KA+1:POKE 835,KA-INT(KA/256)*256:PO
    KE 836,INT(KA/256)                          <253>
725 PRINT"{CLR}PO32768,0:PO43,1:PO44,128:P
    O45,PE(835):POKE46,PE(836):PO55,0:PO56
    ,160                                        <141>
730 PRINT"{HOME,4DOWN}CLR                       <206>
735 PRINT"{HOME,7DOWN}LIST"                     <133>
740 POKE 631,19:POKE 632,13:POKE 633,13:PO
    KE 634,13:POKE 198,4:END                    <087>
745 :                                           <213>
750 :                                           <218>
755 REM *** BMG/EDITOR ***                      <191>
760 GOSUB 1300                                  <000>
765 XD=16:YD=44:X1=0:Y1=0                       <060>
770 X=XD:Y=YD:K=0:POKE V+1,Y:Z=(X>255):POK
    E V,X+(Z*256):POKE 53264,Z*(-1)             <164>
775 POKE V+2,0:POKE V+3,0:POKE V+21,3           <010>
780 GET A$:IF A$=""THEN 780                     <085>
785 A=1*(A$="{DOWN}")+2*(A$="{UP}")+3*(A$=
    "{RIGHT}")+4*(A$="{LEFT}")+5*(A$="{HOM
    E}")+6*(A$="{CLR}"):A=A*-1                  <172>
790 IF A=1 THEN Y=Y+8:IF Y>236 THEN Y=236:
    GOTO 880                                    <034>
795 IF A=2 THEN Y=Y-8:IF Y<44 THEN Y=44:GO
    TO 880                                      <174>
800 IF A=3 AND X=328 AND Y=236 THEN 880         <120>
805 IF A=3 THEN X=X+8:IF X>328 THEN X=16:A
    =1:GOTO 790                                 <074>
810 IF A=4 AND X=16 AND Y=44 THEN 880           <035>
815 IF A=4 THEN X=X-8:IF X<16 THEN X=328:A
    =2:GOTO 795                                 <198>
820 IF A=5 AND K=0 THEN PRINT"{HOME}";:GOT
    O 765                                       <037>
825 IF A=6 AND K=0 THEN PRINT"{CLR}";:GOTO
     765                                        <090>
830 IF A=0 AND K=0 THEN 930                     <239>
835 GOTO 880                                    <185>
840 REM ZEICHEN SCHREIBEN                       <223>
845 IF A$=CHR$(13)THEN 880                      <054>
850 IF A$=CHR$(34)THEN 780                      <047>
855 A=ASC(A$):B=A-161-33*(A<255)-64*(A<192
    )-32*(A<160)+32*(A<96)-64*(A<64)            <203>
860 IF RV=1 THEN B=B+128                        <104>
865 POKE 1024+Y1*40+X1,B                        <189>
870 POKE 55296+Y1*40+X1,PEEK(V+39):A=3:GOT
    O 800                                       <207>
875 REM SPRITES KOORDINIEREN                    <061>
880 IF K=1 THEN 905                             <110>
885 Z=(X>255):POKE V,X+(Z*256):POKE 53264,
    Z*(-1):POKE V+1,Y                           <083>
890 X1=(X-16)/8:Y1=(Y-44)/8                     <208>
895 IF LN=1 THEN K=1                            <226>
900 GOTO 780                                    <242>
905 Z=(X>255):POKE V+2,X+(Z*256):POKE 5326
    4,PEEK(53264)AND 1 OR(Z*(-2)):POKE V+3
    ,Y                                          <112>
910 IF A$=CHR$(13)THEN X2=(X-16)/8:Y2=(Y-4
    4)/8:IF X1<>X2 AND Y1<>Y2 THEN 780          <218>
915 IF A$=CHR$(13)AND LN=1 THEN 1155            <192>
920 GOTO 780                                    <006>
925 REM ZUSATZFUNKTIONEN                        <225>
930 A=1*(A$="{RVSON}")+2*(A$="{RVOFF}")+3*
    (A$="{CTRL-X}")+4*(A$="{CTRL-Y}")+5*(A
    $="{CTRL-Z}")                               <131>
935 A=A+6*(A$="{F1}")+7*(A$="{F3}")+8*(A$=
    "{F5}")+9*(A$="{F7}")+10*(A$=CHR$(148)
    )                                           <089>
940 A=A+11*(A$=CHR$(20))+12*(A$="{CTRL-L}"
    )+13*(A$="{CTRL-F}"):A=A*-1:IF A=0 THE
    N 990                                       <206>
945 REM GONG                                    <189>
950 POKE 54296,15:POKE 54277,0:POKE 54278,
    247:POKE 54276,17:POKE 54273,20             <162>
955 POKE 54272,0:FOR I=1 TO 100:NEXT:POKE
    54276,16                                    <233>
960 IF A=11 THEN GOSUB 1260:A=4:GOTO 810        <113>
965 IF A=12 THEN 1150                           <155>
970 IF A=13 THEN 1285                           <022>
975 ON A GOSUB 1015,1020,1110,1125,1030,11
    95,1210,1220,1225,1235                      <056>
980 GOTO 780                                    <066>
985 REM CURSORFARBEN                            <037>
990 A=1*(A$="{BLACK}")+2*(A$="{WHITE}")+3*
    (A$="{RED}")+4*(A$="{CYAN}")+5*(A$="{P
    URPLE}")+6*(A$="{GREEN}")                   <158>
995 A=A+7*(A$="{BLUE}")+8*(A$="{YELLOW}")+
    9*(A$="{ORANGE}")+10*(A$="{BROWN}")+11
    *(A$="{LIG.RED}")+12*(A$="{GREY 1}")        <172>
1000 A=A+13*(A$="{GREY 2}")+14*(A$="{LIG.G
     REEN}")+15*(A$="{LIG.BLUE}")+16*(A$="
     {GREY 3}"):A=A*-1                          <032>
1005 IF A=0 THEN 845                            <255>
1010 PRINT CO$(A);:POKE V+39,A-1:POKE V+40
     ,A-1:GOTO 780                              <074>
1015 RV=1:RETURN:REM RVS/ON                     <134>
1020 RV=0:RETURN:REM RVS/OFF                    <173>
1025 REM ZEILE ZENTRIEREN                       <179>
1030 ZE$="":Z=Y1:I=0:C1=0:C2=0:ZZ=0             <207>
1035 IF I<40 THEN CH=PEEK(1024+Z*40+I):IF
     CH=32 OR CH=96 THEN C1=C1+1:I=I+1:GOT
     O 1035                                     <134>
1040 I=39                                       <044>
1045 IF I>-1 THEN CH=PEEK(1024+Z*40+I):IF
     CH=32 OR CH=96 THEN C2=C2+1:I=I-1:GOT
     O 1045                                     <173>
1050 IF C1=C2 THEN 780                          <050>
1055 FOR I=C1 TO 40-C2:ZE$=ZE$+CHR$(PEEK(1
     024+Z*40+I))+CHR$(PEEK(55296+Z*40+I))
     :NEXT                                      <161>
1060 IF C1>C2 THEN 1085                         <131>
1065 C=(C2-C1)/2                                <247>
1070 GOSUB 1110                                 <058>
1075 FOR I=C1+C TO 40-C2+C:ZZ=ZZ+2:POKE 10
     24+Z*40+I,ASC(MID$(ZE$,ZZ-1,1))            <006>
1080 POKE 55296+Z*40+I,ASC(MID$(ZE$,ZZ,1))
     :NEXT:GOTO 1105                            <184>
```

Listing 1. Der komfortable Maskengenerator (Fortsetzung)

```
1085 C=(C1-C2)/2                                  <205>
1090 GOSUB 1110                                   <078>
1095 ZZ=0:FOR I=C1-C TO 40-C2-C:ZZ=ZZ+2:PO
     KE 1024+Z*40+I,ASC(MID$(ZE$,ZZ-1,1))         <047>
1100 POKE 55296+Z*40+I,ASC(MID$(ZE$,ZZ,1))
     :NEXT                                        <137>
1105 RETURN                                       <147>
1110 REM ZEILE LOESCHEN                           <238>
1115 PRINT"{HOME}";LEFT$(D$,Y1);S$;"{HOME}
     ":POKE 1024+Y1*40+39,32                      <111>
1120 RETURN                                       <162>
1125 REM SPALTE LOESCHEN                          <146>
1130 IF X1=39 THEN FOR I=0 TO 24:POKE 1024
     +I*40+39,32:NEXT:GOTO 1140                   <117>
1135 PRINT"{HOME}";SPC(X1);:FOR I=1 TO 24:
     PRINT"{SPACE,DOWN,LEFT}";:NEXT:PRINT"
     {HOME}";:POKE 1024+24*40+X1,32               <039>
1140 RETURN                                       <182>
1145 REM LINIE ZEICHNEN                           <090>
1150 LN=1:GOTO 885                                <072>
1155 XD=X2*8+16:YD=Y2*8+44:XN=X2:YN=Y2            <125>
1160 Z=PEEK(1024+X1+40*Y1):C=PEEK(55296+X1
     +40*Y1)                                      <011>
1165 IF X1>X2 THEN X=X1:X1=X2:X2=X:GOTO 11
     75                                           <057>
1170 IF Y2>Y1 THEN Y=Y1:Y1=Y2:Y2=Y                <062>
1175 IF Y1=Y2 THEN FOR I=X1 TO X2:POKE 102
     4+40*Y1+I,Z:POKE 55296+40*Y1+I,C:NEXT
     :GOTO 1185                                   <010>
1180 FOR I=Y2 TO Y1:POKE 55296+40*I+X1,C:P
     OKE 1024+40*I+X1,Z:NEXT                      <006>
1185 X1=XN:Y1=YN:LN=0:K=0:GOTO 770                <105>
1190 REM GROUND+1                                 <014>
1195 G=PEEK(53281)AND 15:IF G<15 THEN G=G+
     1:POKE 53281,G:RETURN                        <195>
1200 IF G=15 THEN G=0:POKE 53281,G:RETURN         <202>
1205 REM BORDER+1                                 <106>
1210 BO=PEEK(53280)AND 15:IF BO<15 THEN BO
     =BO+1:POKE 53280,BO:RETURN                   <164>
1215 IF BO=15 THEN BO=0:POKE 53280,BO:RETU
     RN                                           <125>
1220 POKE 650,255:RETURN:REM REPEAT/ON            <012>
1225 POKE 650,0:RETURN:REM REPEAT/OFF             <229>
1230 REM INSERT                                   <122>
1235 FOR I=39 TO X1 STEP-1:POKE 55296+Y1*4
     0+I,PEEK(55296+Y1*40+I-1)                    <052>
1240 POKE 1024+Y1*40+I,PEEK(1024+Y1*40+I-1
     ):NEXT                                       <109>
1245 POKE 1024+Y1*40+X1,32                        <229>
1250 RETURN                                       <036>
1255 REM DELETE                                   <236>
1260 FOR I=X1 TO 39:POKE 55296+Y1*40+I,PEE
     K(55296+Y1*40+I+1)                           <143>
1265 POKE 1024+Y1*40+I,PEEK(1024+Y1*40+I+1
     ):NEXT                                       <070>
1270 POKE 1024+Y1*40+39,32                        <184>
1275 RETURN                                       <061>
1280 REM EDITOR-ENDE                              <186>
1285 FOR I=15 TO 0 STEP-1:POKE 53281,I:POK
     E 53280,I:NEXT:POKE 49154,G:POKE 4915
     5,BO                                         <124>
1290 POKE 53281,G:POKE 53280,BO                   <004>
1295 POKE V+21,0:GOTO 105                         <119>
1300 REM INIT/SPRITE-INIT                         <242>
1305 GOSUB 1365                                   <040>
1310 D$="{24DOWN}"                                <075>
1315 S$="{39SPACE}"                               <146>
1320 FOR I=832 TO 839:POKE I,0:NEXT               <237>
1325 DIM CO$(16):CO$="{BLACK,WHITE,RED,CYA
     N,PURPLE,GREEN,BLUE,YELLOW,ORANGE,BRO
     WN,LIG.RED,GREY 1,GREY 2,LIG.GREEN,LI
     G.BLUE,GREY 3}":FOR I=1 TO LEN(CO$):C
     O$(I)=MID$(CO$,I,1):NEXT                     <237>
1330 FOR I=704 TO 704+63:READ Q:POKE I,Q:N
     EXT:POKE 2040,11:POKE 2041,11                <205>
1335 V=53248:POKE V+39,15:POKE V+40,15            <062>
1340 RETURN                                       <128>
1345 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,
     255,128,1,0,128,1,0,128,1,0,128              <156>
1350 DATA 1,0,128,1,0,128,1,0,128,1,0,128,
     1,0,128,1,255,128,0,0,0,0,0,0,0              <106>
1355 DATA 0,0,0,0,0,0,0,0,0,0,0,0                 <171>
1360 DATA"{BLACK}","{WHITE}","{RED}","{CYA
     N}","{PURPLE}","{GREEN}","{BLUE}","{Y
     ELLOW}","{ORANGE}","{BROWN}","{LIG.RE
     D}","{GREY 1}","{GREY 2}","{LIG.GREEN
     }","{LIG.BLUE}","{GREY 3}"                   <215>
1365 G=11:BO=G:POKE 53281,G:POKE 53280,BO         <110>
1370 PRINT"{CLR,3DOWN,6SPACE,RVSON,GREY 3,
     6SPACE,RVOFF,3SPACE,RVSON,3SPACE,RVOF
     F,3SPACE,RVSON,3SPACE,RVOFF,2SPACE,RV
     SON,7SPACE,RVOFF}                            <207>
1375 PRINT"{6SPACE,RVSON,SPACE,BLACK,SPACE
     ,GREY 3,SPACE,BLACK,3SPACE,GREY 3,SPA
     CE,RVOFF,2SPACE,RVSON,SPACE,BLACK,2SP
     ACE,GREY 3,SPACE,RVOFF,SPACE,RVSON,SP
     ACE,BLACK,2SPACE,GREY 3,SPACE,BLACK,S
     PACE,RVOFF,SPACE,RVSON,GREY 3,SPACE,B
     LACK,SPACE,GREY 3,SPACE,BLACK,3SPACE,
     GREY 3,SPACE,BLACK,SPACE,RVOFF}              <057>
1380 PRINT"{6SPACE,RVSON,GREY 3,SPACE,BLAC
     K,SPACE,GREY 3,SPACE,BLACK,SPACE,RVOF
     F,2SPACE,RVSON,GREY 3,SPACE,BLACK,SPA
     CE,RVOFF,SPACE,RVSON,GREY 3,SPACE,BLA
     CK,SPACE,RVOFF,2SPACE,RVSON,GREY 3,SP
     ACE,BLACK,2SPACE,RVOFF,SPACE,RVSON,GR
     EY 3,SPACE,BLACK,SPACE,RVOFF,SPACE,RV
     SON,GREY 3,SPACE,BLACK,SPACE,GREY 3,S
     PACE,BLACK,SPACE,RVOFF,2SPACE,RVSON,2
     SPACE,RVOFF}                                 <109>
1385 PRINT"{6SPACE,RVSON,GREY 3,6SPACE,BLA
     CK,2SPACE,RVOFF,SPACE,RVSON,GREY 3,SP
     ACE,BLACK,SPACE,GREY 3,SPACE,RVOFF,SP
     ACE,RVSON,BLACK,2SPACE,GREY 3,SPACE,R
     VOFF,SPACE,RVSON,SPACE,BLACK,SPACE,RV
     OFF,SPACE,RVSON,GREY 3,SPACE,BLACK,SP
     ACE,GREY 3,SPACE,BLACK,SPACE,GREY 3,3
     SPACE,RVOFF}                                 <208>
1390 PRINT"{6SPACE,RVSON,SPACE,BLACK,SPACE
     ,GREY 3,SPACE,BLACK,3SPACE,GREY 3,SPA
     CE,RVOFF,2SPACE,RVSON,SPACE,BLACK,SPA
     CE,GREY 3,2SPACE,RVOFF,SPACE,RVSON,2S
     PACE,BLACK,SPACE,GREY 3,SPACE,BLACK,S
     PACE,RVOFF,SPACE}";                          <125>
1391 PRINT"{RVSON,GREY 3,SPACE,BLACK,SPACE
     ,GREY 3,3SPACE,BLACK,SPACE,GREY 3,SPA
     CE,BLACK,SPACE,RVOFF}                        <018>
1395 PRINT"{6SPACE,RVSON,GREY 3,SPACE,BLAC
     K,SPACE,GREY 3,SPACE,BLACK,SPACE,RVOF
     F,2SPACE,RVSON,GREY 3,SPACE,BLACK,SPA
     CE,RVOFF,SPACE,RVSON,GREY 3,SPACE,BLA
     CK,SPACE,GREY 3,SPACE,BLACK,SPACE,GRE
     Y 3,SPACE,BLACK,SPACE,GREY 3,SPACE,BL
     ACK,SPACE,GREY 3,SPACE,BLACK,SPACE,RV
     OFF,SPACE,RVSON,GREY 3,SPACE,BLACK,5S
     PACE,GREY 3,SPACE,BLACK,SPACE,RVOFF}         <018>
1400 PRINT"{6SPACE,RVSON,GREY 3,6SPACE,BLA
     CK,2SPACE,RVOFF,SPACE,RVSON,GREY 3,3S
     PACE,BLACK,SPACE,RVOFF,SPACE,RVSON,SP
     ACE,GREY 3,3SPACE,BLACK,SPACE,RVOFF,S
     PACE,RVSON,GREY 3,7SPACE,BLACK,SPACE,
     RVOFF}                                       <057>
1405 PRINT"{7SPACE,RVSON,6SPACE,RVOFF,3SPA
     CE,RVSON,3SPACE,RVOFF,3SPACE,RVSON,3S
     PACE,RVOFF,2SPACE,RVSON,7SPACE,RVOFF}        <081>
1410 PRINT"{2DOWN,6SPACE}B{GREY 3}ILDSCHIR
     M{YELLOW}-{BLACK}M{GREY 3}ASKEN{YELLO
     W}-{BLACK}G{GREY 3}ENERATOR                  <127>
1415 N$=CHR$(82)+CHR$(79)+CHR$(77)+CHR$(65
     )+CHR$(78)+CHR$(32)+CHR$(66)+CHR$(65)        <099>
1420 N$=N$+CHR$(82)+CHR$(84)+CHR$(75)+CHR$
     (69)                                         <233>
1425 PRINT"{DOWN,4SPACE,YELLOW}WRITTEN 3/1
     986{2SPACE}BY ";N$                           <146>
1430 POKE 2,0:PRINT"{3DOWN}"SPC(8)"{GREY 3
     }SOLLEN 'SHIFT/SPACE' DEN"                   <117>
1435 PRINT"{4SPACE}'SPACE'-ZEICHEN GLEICH
     SEIN ? J{3LEFT}";:INPUT A$                   <200>
1440 IF A$<>"N"THEN POKE 2,1                      <247>
1445 RETURN                                       <233>
```

Listing 1. »Maske« (Schluß)

AUFLÖSUNG

DER KNOBEL ECKE 1

Unsere erste Knobelecke war ein voller Erfolg. Unsere Leser haben sich bei der Lösung der nicht ganz einfachen Aufgabe wieder einmal selbst übertroffen. Wir stellen Ihnen die drei Sieger mit ihren Listings vor.

Im Sonderheft 40 fanden Sie zum ersten Mal die »Knobelecke«. Hier konnten alle Basic-Programmierer ihr Können beweisen: Es ging darum, ein Programm zu schreiben, das in kürzester Zeit eine bestimmte logische Aufgabe löst. Für diejenigen, die diese Ausgabe nicht besitzen, hier nur ganz kurz noch einmal die Aufgabenstellung:

Fünf Freunde haben einen Schatz gefunden. Nacheinander wirft nun jeder der fünf zuerst eine bestimmte Anzahl Münzen in einen Fluß (der erste acht, der zweite neun, und so weiter). Die jeweils verbleibende Anzahl läßt sich immer genau (ohne Rest) durch fünf teilen. Dieser fünfte Teil wird im Gepäck der jeweiligen Person versteckt, worauf die nächste Person wieder erst Münzen in den Fluß wirft und sich dann ihren Anteil nimmt.

Die daraufhin verbleibende Anzahl n wird wieder genau durch fünf geteilt. Die Aufgabe: Wie viele Münzen waren ganz am Anfang vorhanden, wie viele nach der Verteilung (Anzahl n)? Ein Basic-Programm sollte beide Werte möglichst schnell berechnen und ausgeben.

Das Echo auf diese Aufgabe war überwältigend. Da wir natürlich nicht mit der Stoppuhr die Zeiten nehmen konnten und sogar die im Computer eingebaute Uhr TI für diesen Zweck zu ungenau war, mußten wir mit dem Laufzeit-Meß-System von Franz Stoiber aus der 64'er 4/86 die Zeiten auf den zehnten Teil einer Millisekunde genau messen, zu sehr ähneln sich die Laufzeiten der Konkurrenten.

Der dritte Sieger

So ergaben sich drei Gewinner, deren Programme sich in der Laufzeit nur ganz knapp unterscheiden. Wir wollen Ihnen die Sieger vorstellen (siehe auch Tabelle 1 und Textkasten).

Das Programm von Erich Wollenberg aus Filsum benötigt 237,4 ms, um die richtigen Ergebnisse **9363** und **3040** zu berechnen (Listing 1). Es wurde aus der folgenden Formel entwickelt, aus der die Aufgabe direkt zu ersehen ist:

```
z = ((((20+12)*f+11)*f+10)*f+9)*f+8
```

z ist die Anzahl der gefundenen Münzen und f die Anzahl, die am »Morgen danach« noch vorhanden war.

Damit die Forderung nach der ganzzahligen Teilbarkeit erfüllt ist, müssen die Klammerausdrücke vom Typ Integer sein.

Die Formel führte zunächst zu folgendem Versuchsprogramm:

```
 10 TI$="000000"
 20 F=5/4:V=20
 30 I=I+V:Z=I*F
 40 Z=(Z+12)*F:IF Z>INT(Z)THEN30
 50 Z=(Z+11)*F:IF Z>INT(Z)THEN30
 60 Z=(Z+10)*F:IF Z>INT(Z)THEN30
 70 Z=(Z+ 9)*F:IF Z>INT(Z)THEN30
 80 Z=Z+8
 90 PRINT Z,I
100 PRINT TI/60"SEKUNDEN"
```

Nun wird die Zeile 80 in 90 untergebracht und die Zeile 40 auf die Zeilen 30 und 50 verteilt (Änderungen sind *kursiv* gedruckt):

```
 10 TI$="000000"
 20 F=5/4:V=20
 30 I=I+V:Z=I*F*F:IF Z>INT(Z)THEN30
 50 Z=(Z+26)*F:IF Z>INT(Z)THEN30
 60 Z=(Z+10)*F:IF Z>INT(Z)THEN30
 70 Z=(Z+ 9)*F:IF Z>INT(Z)THEN30
 90 PRINT Z+8,I
100 PRINT TI/60"SEKUNDEN"
```

Um auf die Multiplikation in Zeile 30 nach jeder IF-Abfrage verzichten zu können, lassen wir sie nur einmal in Zeile 20 ausführen. In Zeile 90 muß dann der Ausdruck I durch F*F geteilt werden:

```
 10 TI$="000000"
 20 F=5/4:V=20*F*F
 30 I=I+V:IF I>INT(I) THEN30
 50 Z=(I+26)*F:IF Z>INT(Z)THEN30
 60 Z=(Z+10)*F:IF Z>INT(Z)THEN30
 70 Z=(Z+ 9)*F:IF Z>INT(Z)THEN30
 90 PRINT Z+8,I/(F*F)
100 PRINT TI/60"SEKUNDEN"
```

Der Term 20*F*F (F = 5/4) wird erst geradzahlig, wenn er mit 4 multipliziert wird. Also: 80*F*F. Dies wird in Zeile 20 übernommen. In den Zeilen 60 und 70 muß V wiederum

mit 4 beziehungsweise 16 multipliziert werden. Zusätzlich muß in Zeile 20 eine neue Variable für die Multiplikation von V vorgesehen werden.

```
 10 TI$="000000"
 20 F=5/4:V=80*F*F:T=V
 30 I=I+V:IF I>INT(I)THEN30
 50 Z=(I+26)*F:IF Z>INT(Z)THEN30
 60 Z=(Z+10)*F:IF Z>INT(Z)THENV=T* 4:GOTO30
 70 Z=(Z+ 9)*F:IF Z>INT(Z)THENV=T*16:GOTO30
 90 PRINT Z+8,I/(F*F)
100 PRINT TI/60"SEKUNDEN"
```

Die Zeilen 30 und 50 werden zu einer einzigen Zeile 30 zusammengefaßt:

```
 10 TI$="000000"
 20 F=5/4:V=80*F*F:T=V
 30 I=I+V:Z=(I+26)*F:IF Z>INT(Z) THEN30
 60 Z=(Z+10)*F:IF Z>INT(Z)THEN V=T* 4:GOTO30
 70 Z=(Z+ 9)*F:IF Z>INT(Z)THEN V=T*16:GOTO30
 90 PRINT Z+8,I/(F*F)
100 PRINT TI/60"SEKUNDEN"
```

Das wäre auch schon das komplette Listing (Listing 1). Es wurden lediglich die Zeilen neu durchnumeriert. Die Laufzeit beträgt, wie gesagt, 237,4 Millisekunden.

Der zweite Sieger

Etwas schneller ist da schon das Programm von Christian Geist aus dem Ort mit dem schönen Namen Linsengericht. 211,1 ms nach dem Start erscheint bereits wieder das »READY.« auf dem Bildschirm. Das Ausgangsprogramm ist das folgende:

```
 10 TI$="000000"
 20 Z=5
 30 X=Z
 40 FOR Y=12TO8STEP-1
 50 X=5/4*X+Y
 60 IF X<>INT(X)THEN Z=Z+5:GOTO30
 70 NEXT
 80 PRINT"A)";X
 90 PRINT"B)";Y
100 PRINT TI$
```

Der Computer geht also alle Möglichkeiten durch, wie viele Münzen am nächsten Morgen übrig sein könnten, und rechnet dann zurück, wie viele Münzen insgesamt gefunden wurden. In Zeile 50 kann die Gleichung etwas umgestellt werden:

`Z=X-Y-(X-Y)/5 <=>X=5/4*Z+Y.`

Hierbei bezeichnet X den bisherigen Rest, Y die Stunde und Z den neuen Rest. Da der C64 rückwärts rechnet und den neuen Rest Z nicht mehr benötigt, kann die Formel rekursiv gestaltet werden, für Z setzen wir X ein. In Zeile 60 werden alle Zahlen aussortiert, die nicht natürlich (sondern Kommazahlen) sind. Der Computer nimmt dann fünf Münzen mehr an als in der letzten Rechnung und fängt wieder von vorn an. Die erste Version des Programmes lief noch etwa 17 Sekunden (siehe Tabelle 2).

Um nach der ersten Rechnung eine natürliche Zahl zu erhalten, muß der Rest am Morgen durch vier teilbar sein. Es gibt jetzt zwei Bedingungen für X: die Teilbarkeit durch 5 und durch 4. Zusammengefaßt muß X also durch 20 teilbar sein. Wir ändern das Programm wie folgt:

```
20 Z=20
60 IFX<>INT(X)THENZ=Z+20:GOTO30
```

DIE 3 GE



Klaus Kursawe

1.

```
0 TI$="000000":C=5:B=3125                      <091>
1 B=B-4:A=B-8:A=A-A/C:IF A<>INT(A)GOTO 1       <072>
2 A=B-1:D%=5-(A/C-INT(A/C))*C+.5:B=3125*D%     <117>
3 B=B-4:A=B-8:A=A-A/C:IF A<>INT(A)GOTO 3       <075>
4 PRINT B-(D%*2101+20);B;TI/60                 <228>
```

Listing 3. Das Siegerprogramm schrieb K. Kursawe

Auch unser Gewinner **Klaus Kursawe** aus 2844 Lemförde ist noch Schüler. Er ist 16 Jahre alt und besucht zur Zeit noch die zehnte Klasse des Gymnasiums. Seinen Commodore 64 bekam er zusammen mit seiner Schwester vor vier Jahren. Er ist begeisterter Hobbyprogrammierer und arbeitet daher auch heute noch an diesem Computer. An der notwendigen Übung scheint es ihm in keinster Weise zu fehlen, sein glänzender erster Platz in unserer ersten Knobelecke dürfte Beweis genug sein. Er darf sich auf drei Bücher aus dem Markt & Technik Verlag freuen.

Das gesamte Team der 64'er-Sonderhefte gratuliert den Gewinnern auf diesem Weg ganz herzlich und bedankt sich bei allen, die mitgemacht haben.

Rang	Gewinner	Alter	Laufzeit
1	Klaus Kursawe	16	207,7 ms
2	Christian Geist	17	211,1 ms
3	Erich Wollenberg	36	237,4 ms

Tabelle 1. So wurde gewonnen

Tabelle 2. Laufzeiten der verschiedenen Versionen des zweiten Platzes (nach Angaben des Autors)

Version	Laufzeit
1	17 Sekunden
2	6 Sekunden
3	2,23 Sekunden
4	1,27 Sekunden
5	0,8 Sekunden
6	0,63 Sekunden
7	0,55 Sekunden
8	0,53 Sekunden
9	0,52 Sekunden
10	0,22 Sekunden
11	0,17 Sekunden

Aus 17 Sekunden Laufzeit werden so knapp 6 Sekunden. X muß vor der zweiten Rechnung auch durch vier teilbar sein, damit man einen natürlichen (ganzzahligen) Wert erhält. In der Gleichung Z=5/4*X+Y ist bei der ersten Rechnung Y=12. Damit sind also Z und Y jeweils durch vier teilbar, so daß dies auch für 5/4*X gelten muß. Da man nur eine natürliche Zahl erhält, wenn X durch 4 teilbar ist, muß X durch 16 teilbar sein. Es muß also insgesamt durch 80 teilbar sein, da die Division durch fünf auch ganzzahlig sein soll. Ändern wir das Programm:

```
20 Z=80
60 IFX<>INT(X)THENZ=Z+80:GOTO30
```

WINNER

2.



Christian Geist

```
1 TI$="000000":Y=480:X=Y:Z=X:W=5/4:V=12:U=
  8:T=-1:S=1280                          <142>
2 FOR Y=V TO U STEP T:X=X*W+Y:NEXT       <249>
3 IF X<>INT(X)THEN Z=Z+S:X=Z:GOTO 2      <174>
4 PRINT"A)";X,"B)";Z,TI$,TI              <172>
```

Listing 2. Unser zweiter Sieger stammt von C. Geist

Christian Geist aus 6464 Linsengericht 2 wurde am 29.2.1972 in Gelnhausen geboren und besucht zur Zeit die elfte Klasse am Gymnasium. Vor etwa neun Monaten kaufte er sich den C64 als ersten Computer. Dieser war eigentlich nur als Ergänzung zum Informatik-Unterricht gedacht, doch schon nach kurzer Zeit verbrachte er immer mehr Zeit mit dem Rechner. Dabei interessiert er sich weniger für die Spiele, die natürlich auch dazugehören, als vielmehr für Probleme und Aufgaben, wie etwa die Knobelecke. Das 64'er-Magazin lernte er bei seinem Onkel kennen. Bald merkte er, daß diese Zeitschrift die ideale Ergänzung zum Computer ist. Seit Januar 1989 liest er das Stammheft und die Sonderhefte regelmäßig. Da stieß er auf die Knobelecke und beschloß, mitzumachen. Es hat sich für ihn gelohnt: Sein Programm kam auf den zweiten Platz und bescherte ihm zwei Bücher.





Erich Wollenberg

```
10 TI$="000000"                          <245>
20 F=5/4:V=80*F*F:T=V                    <078>
30 I=I+V:Z=(I+26)*F:IF Z>INT(Z)THEN 30   <235>
40 Z=(Z+10)*F:IF Z>INT(Z)THEN V=T*4:GOTO 3
   0                                     <111>
60 Z=(Z+9)*F:IF Z>INT(Z)THEN V=T*16:GOTO 3
   0                                     <239>
90 PRINT Z+8,I/(F*F)                     <246>
100 PRINT T/60"SEKUNDEN                  <193>
```

Listing 1. E. Wollenberg erreicht den dritten Preis

Erich Wollenberg aus 2951 Filsum ist 36 Jahre alt. Er verdient seine Brötchen als Beamter im mittleren fernmeldetechnischen Dienst der Deutschen Bundespost. Im Herbst letzten Jahres hat er sich einen C64 gekauft, der in der Hauptsache von seinem Sohn als Spielcomputer genutzt wird. Da mathematische Probleme schon immer sein Interesse geweckt haben, hat er sich sofort daran gemacht, eine Lösung für die gestellte Aufgabe zu finden. Nachdem die Formel gefunden war, mußte das Versuchsprogramm geschrieben werden. Es wurde im wesentlichen nicht geändert, jedoch mit einigen Vereinfachungen versehen und von unnötigen Rechenschritten befreit. Es belegt einen stolzen dritten Platz, Herr Wollenberg erhält dafür ein Markt&Technik-Buch.

Diese Verbesserung bringt einen Gewinn von 4 Sekunden, die Laufzeit beträgt jetzt 2 Sekunden. Die erste Rechnung lautet:

```
Z=5/4*X+12
```

und die zweite Rechnung:

```
W=5/4*Z+11
```

W muß wieder durch vier teilbar sein. Da 11 geteilt durch vier den Rest drei ergibt, muß 5/4*X um eins größer sein als eine durch vier teilbare Zahl. Allgemein also:

```
5/4*Z=A*4+1
```

Dabei ist A eine natürliche Zahl. Durch Umstellen, Einsetzen und erneutes Umstellen erhält man:

```
A=24/64*X+X/64+0,5
```

Daraus ist ersichtlich, daß X durch 32 teilbar sein muß, da sonst A keine natürliche Zahl ist. Weiterhin muß X durch fünf teilbar sein, insgesamt muß also Teilbarkeit durch 160 gegeben sein:

```
20 Z=160
60 IFX<>INT(X)THENZ=Z+160:GOTO30
```

Diese einfache Verbesserung bringt uns auf eine Zeit von etwas über einer Sekunde. Aus der letzten Gleichung ersieht man außerdem, daß X nicht durch 64 teilbar sein darf. Da jedes Produkt von 160 und einem geraden Faktor automatisch durch 64 teilbar ist, können wir mit Z=Z+320 diese Werte umgehen:

```
60 IFX<>INT(X)THENZ=Z+320:GOTO30
```

Zeile 20 wird diesmal nicht geändert. Bekanntlich wird ein Basicprogramm schneller, wenn man alle Konstanten am Anfang in Variablen definiert, da der Interpreter auf diese schneller zugreift als auf Dezimalzahlen. Ändern wir also Zeile 10:

```
10 TI$="000000":X=160:Z=X:W=5/4:V=12:U=8:T=-1:S=320
20 entfällt
30 entfällt
40 FOR Y=V TO U STEP T
50 X=W*X+Y
60 IF X<>INT(X)THEN Z=Z+S:X=Z:GOTO40
```

Nun fassen wir noch einige Zeilen zusammen:

```
40 FOR Y=V TO U STEP T:X=W*X+Y:IF X<>INT(X)
   THEN Z=Z+S:X=Z:GOTO40
50 entfällt
60 entfällt
70 NEXT:PRINT"A)";X,"B)";Z,TI$,TI
80 entfällt
90 entfällt
100 entfällt
```

Als nächstes definierten wir den Term Y+C vor und ersetzen GOTO40 durch Y=C:NEXT

```
10 TI$="000000": Y=1:X=160:Z=X:W=5/4:V=12:U=8:T=
   -1:S=320:C=13
20 FOR Y=V TO U STEP T:X=W*X+Y:IF X<>INT(X)
   THEN Z=Z+S:X=Z:Y=C:NEXT
```

Nun soll in Zeile 10 die Variable Y von einem sinnvolleren Wert (160) vorbelegt werden. Außerdem setzen wir X=Y.

```
10 TI$="000000":Y=160:X=Y:Z=Y:W=5/4:V=12:U=8:T=
   -1:S=320:C=13
```

Diese neunte Version läuft in 31/60 Sekunden. Jetzt verbessern wir die drei Rechnungen:

```
1. Z=5/4*X+12
2. W=5/4*Z+11
3. V=5/4*W+10
   W=A*4+2 (abgeleitet von 3.)
```

Das ist im Prinzip wieder dieselbe Überlegung wie oben. Die Endgleichung lautet diesmal:

```
A-7=15*X/32+5*X/256+5/8
```

Da X durch 32 teilbar ist, ergibt 15*X/32 eine ganze Zahl. Außerdem läßt sich der Bruch 5*X/256 auf einen Bruch mit 8 im Nenner kürzen. Damit haben die Brüche 5*X/256 und 5/8 nach Einsetzen von X den gleichen Nenner und lassen

sich problemlos addieren. Als Ergebnis muß eine durch acht teilbare Zahl herauskommen. Den entsprechenden X-Wert erhält man nach der Gleichung X=7+8*B, wobei B eine natürliche Zahl ist. Diese Werte sind noch nicht durch acht teilbar, wir erweitern die Formel also mit acht: X=224+256*B. Dieser Term erfüllt aber immer noch nicht alle Forderungen, denn X muß auch durch fünf teilbar sein. Dies ist nur der Fall, wenn die Einerstelle von Y eins oder sechs ist. Diese Regelmäßigkeit können wir nutzen: da B mindestens 1 ist, kann man zu der 224 einmal 256 addieren. Jetzt wird zu diesem Wert jeweils 5*256 addiert, somit erhält man eine Anzahl Münzen, die alle Bedingungen erfüllt. Mit dieser Anzahl kann der Computer wieder zurückrechnen. Außerdem numerieren wir die Zeilen neu durch (bringt keinen Geschwindigkeitsvorteil):

```
1 TI$="000000":Y=480:X=Y:Z=Y:W=5/4:V=12:U=8:T=
  -1:S=1280:C=13
```

Diese Version benötigt $^{13}/_{60}$ Sekunden zur Berechnung. Etwas schneller wird unser Programm noch, wenn die IF-Abfrage hinter die Schleife verlagert wird:

```
(C=13 am Ende von Zeile 1 entfällt)
2 FOR Y=V TO U STEP T:X=W*X+Y:NEXT
3 IF X<>INT(X)THEN Z=Z+S:X=Z:GOTO2
4 PRINT"A)";X,"B)";Y,TI$,TI
```

Werden in Zeile 2 die beiden Faktoren vertauscht, steigert sich die Zeit von $^{11}/_{60}$ auf $^{10}/_{60}$ Sekunden:

```
2 FOR Y=V TO U STEP T:X=X*W+Y :NEXT
```

Damit wären wir auch schon beim fertigen Programm (Listing 2). Man könnte es sogar noch etwas schneller machen, wenn bei der IF-Abfrage statt ungleich auf größer getestet wird (wie in Listing 1, dritter Platz), da der INT-Wert einer Zahl niemals kleiner ist als die Zahl selbst. Auch die Strichpunkte in Zeile 4 könnten zugunsten der Laufzeit entfallen.

Der stolze Gewinner

Da der Programmautor diese Idee jedoch leider nicht selbst hatte, reichte es nicht ganz zum ersten Platz. Den belegt das 0,0034 Sekunden schnellere Programm von Klaus Kursawe aus Lemförde. Tusch: sein Programm (Listing 3) läuft schlappe 207,7 Millisekunden. Auch er kam nach mehreren Vorabversionen zu seinem Siegerprogramm. Die erste Version lautete

```
 5 TI$="000000":F=1:Z=5:G=Z-1
10 B=F*Z*Z*Z*Z*Z
15 A=B
20 A=A- 8:A=A-(A/Z):IF A<>INT(A)GOTO80
30 A=A- 9:A=A-(A/Z):IF A<>INT(A)GOTO80
40 A=A-10:A=A-(A/Z):IF A<>INT(A)GOTO80
50 A=A-11:A=A-(A/Z):IF A<>INT(A)GOTO80
60 A=A-12:A=A-(A/Z):IF A<>INT(A)GOTO80
70 IF A/Z=INT(A/Z)THEN PRINT B,A,TI,TI$:END
75 F=F+1:GOTO10
80 B=B-G:GOTO15
```

Die Grundidee ist die folgende: Wenn man nicht berücksichtigt, daß der Rest (die Münzen, die bei der Endaufteilung noch da sind) durch fünf teilbar ist, so ist die erste Lösung etwas kleiner als 5^5. 5^5 wäre die Lösung ohne weggeworfene Münzen). Das Prinzip des Programmes ist, daß von 5^5 (3125) heruntergezählt wird, bis eine Lösung gefunden wird. Ist diese nicht durch fünf teilbar, so wird von 2mal 5^5 heruntergezählt und so weiter. Die Lösung ist also Xmal 5^5. Die Reihenfolge der Ausgabe ist: Rest, Gesamtsumme, Zeit in $^{1}/_{60}$ Sekunden, Zeit HHMMSS.

Das erste Programm geht wie oben beschrieben vor. Ein Trick wird beim Herunterzählen verwendet: Man muß nicht immer um eins abwärts zählen. Bei n Personen ist die Schrittweite n-1, hier also vier. Dies funktioniert, da 5^5 um eine bestimmte Uhrzeit (fünf Uhr) eine Lösung ist, und die Lösung für jede Stunde, die der erste früher anfängt, um vier größer wird. Beispiel:

Die Startzeit ist 7 Uhr = Lösung von Startzeit + 4.

Nun die zweite Version des Programmes:

```
1 TI$="000000":A=0:C=5:B=A
2 D=D+1:B=3125*D
3 B=B-4:A=B-8
4 A=A-A/C:IFA<>INT(A)GOTO3
5 A=A-9:A=A-A/C-10:A=A-A/C-11:A=A-A/C-12:A=
  A-A/C:IF A/C<>INT(A/C)GOTO2
6 PRINT A;B;TI;TI$:END
```

Im ersten Programm siebt Zeile 20 schon alle falschen Lösungen aus. Deshalb konnten Zeilen 30 bis 60 weggelassen werden. Ferner wurden einige Berechnungen weggelassen. Nun zur dritten Version:

```
  1 TI$="000000":A=0:C=5:B=A
  2 B=3125
  3 B=B-4:A=B-8
  4 A=A-A/C:IF A<>INT(A)GOTO3
  5 A=A-9:A=A-A/C-10:A=A-A/C-11:A=A-A/C-12:A=
    A-A/C: D%=5-(A/C-INT(A/C))*C+.5
  6 B=3125*D%
  7 B=B-4:A=B-8
  8 A=A-A/C:IF A<>INT(A)GOTO7
  9 A=A-9:A=A-A/C-10:A=A-A/C-11:A=A-
    A/C-12:A=A-A/C:
    PRINT A,B,TI
100 REM A=B-8:acht weggeworfene Münzen
```

Hier verwenden wir einen neuen Trick: Aus der Lösung von X=1 wird sofort der richtige X-Wert (D%) ermittelt, wodurch weniger ausprobiert werden muß! Dieser Kniff hebt dieses Programm wohl an die Siegerposition. Doch nun die vierte und letzte Version:

```
0 TI$="000000":C=5:B=3125
1 B=B-4:A=B-8:A=A-A/C:IF A<>INT(A)GOTO1
2 A=B-1:D%=5-(A/C-INT(A/C))*C+.5:B=3125*D%
3 B=B-4:A=B-8:A=A-A/C:IF A<>INT(A)GOTO3
4 PRINT B-(D%*2101+20);B;TI
```

Zeile 5 aus dem dritten Programm wird hier durch A=B-1 ersetzt. B-1 steht für X (eins) mal 2102+20. Da es nur auf den 5er Rest ankommt, kann man auch '1' schreiben. X mal 2102+20 ist eine Konstante, die zusammen mit dem Rest für jede Startzeit die Gesamtsumme ergibt. Hier wurde aus diesem Wert der Rest ermittelt.

Eine kleine Anmerkung sei noch gestattet: Statt von 5^5 herunterzuzählen, kann man auch die Formel B=X*5^5-12 $(= \text{Personen})^{\text{Personen}}$ mal X + (Personen-1)*(Personen-Startzeit)) verwenden. Sie kann zum Beispiel in die Zeilen 1 und 3 von Listing 3 eingesetzt werden, der jeweilige Rest wird dann weggelassen. Dadurch werden die Programme noch etwas kürzer und schneller.

Das waren also die drei Siegerprogramme. Eine Spitzenleistung, finden Sie nicht? Vor allem, wenn man bedenkt, daß unser Probeprogramm immerhin 13 Sekunden brauchte. Sollte es diesmal nicht mit dem Gewinnen geklappt haben, trösten Sie sich: Die nächste Knobelecke kommt bestimmt. Und wer weiß, vielleicht sind Sie dann unter den stolzen Gewinnern? Wenn Sie übrigens Ideen für eine neue Knobelecke haben, so würden wir uns auch über deren Einsendung sehr freuen. (Nikolaus Heusler/ag)

DIE KNOBEL ECKE 2

Eine neue Herausforderung für alle Basic-Knobler: Mit der richtigen Lösung und einer schnellen Zeit warten wieder attraktive Preise auf die Sieger dieser Aufgabe.

Zwei Spitzenspiele, die für Abwechslung nach anstrengender Programmiertätigkeit sorgen, und einen Basic-Compiler der Spitzenklasse (Bild 1) haben wir für die drei Gewinner der neuen Aufgabe ausgesucht. Der Sieger erhält das komplette Paket, der Gewinner des zweiten Preises zwei Disketten nach freier Auswahl. Der dritte Sieger kann sich ein Programm aussuchen.

Die Lösung sollte nicht viele Schwierigkeiten bereiten. Der eigentliche Reiz der Aufgabe liegt wieder darin, einen optimalen Programmalgorithmus zu entwickeln, mit dem nach kürzester Zeit die richtige Lösung auf dem Bildschirm erscheint.

Die Aufgabe

In der Villa »Hammerthal« wird das traditionelle Gartenfest vorbereitet. Alle Angestellten sind emsig damit beschäftigt, den Garten zu schmücken.

Bei einem Rundgang am Tag des Fests stellt der Gastgeber, Graf Bertholdy, mit Entsetzen fest, daß der vorhandene Getränkevorrat kaum reichen wird, seine Gäste ausreichend zu bewirten. Es fehlen vor allem Wein-, Bier- und Sektflaschen. Viel Zeit bleibt jedoch nicht mehr, das Fest beginnt bereits in wenigen Stunden.

Zu allem Unglück befindet sich im ganzen Haus kein Geld. Die Villa liegt weitab von jeder Ortschaft, und es ist Wochenende. Woher also die Getränke besorgen?

Die »Rettung« naht mit Minna, dem Kindermädchen. Minna kennt einen Getränkemarkt ganz in der Nähe, und sie hat genau 100 Mark bei sich.

Ziemlich erleichtert beauftragt Graf Betholdy sie sofort, für diesen Betrag exakt 100 Flaschen an Getränken zu besorgen. Eine Flasche Wein kostet 3 Mark, eine Flasche Sekt 10 Mark, eine Flasche Bier 0,50 Mark.

Helfen Sie Minna, diesen Auftrag zu erledigen. Wie viele Flaschen von jeder Sorte muß sie einkaufen?



Bild 1. Zwei Spiele und ein Basic-Compiler warten auf die drei Gewinner

Die Regeln

Schreiben Sie ein kurzes Basic-Programm, das folgende Bedingungen erfüllen muß:
- Die Zeit muß am Ende ausgegeben werden (mit TI$ oder TI). Das Listing muß mit »TI$ = "000000"« beginnen.
- Assembler-Routinen dürfen nicht verwendet werden.
- Die richtige Lösung muß vor Ausgabe der Zeit auf dem Bildschirm sichtbar sein.

Schicken Sie das Programm bis zum 31.7.1989 an folgende Adresse:

Markt & Technik Verlag AG
Redaktion 64'er-Sonderheft
Stichwort: Knobelecke 2
Hans-Pinsel-Straße 2
8013 Haar bei München

Bitte schicken Sie das Programm nur auf Diskette ein. Das erleichtert uns die schnelle Auswertung.

Haben Sie noch Ideen für weitere Knobeleien? Schicken Sie uns diese bitte an die gleiche Adresse. Die Idee zu dieser Knobelei stammt von Hans-Jürgen Kersten.

Viel Spaß beim Knobeln wünscht die Sonderhefte-Redaktion. (ef)

Rasterinterrupt –

Zweifarbiger Bildschirm(rahmen), Text und Grafik oder zwei verschiedene Zeichensätze gleichzeitig auf dem Bildschirm, Sprites im Bildschirmrahmen – alles kein Problem mit Raster-IRQs. In unserem kleinen Kurs lesen Sie, was das ist und wie sie programmiert werden.

Kein modernes Spiel kommt ohne sie aus. Viele Grafikprogramme verwenden sie. Auch in dem einen oder anderen Utility oder Anwendungsprogramm trifft man auf sie. Die Rede ist von einem »Geheimtip«, wenn es um Grafik geht: den Raster-Interrupts. Sollten Sie noch nicht zu den »Eingeweihten« gehören, die wissen, was man damit alles anstellen kann, so sind Sie hier genau richtig. Dieser Artikel soll Sie in die Geheimnisse der Ra-IRQs (das ist die allgemein gebräuchliche Abkürzung) einweihen. Es ist zwar gar nicht so schwer, diesen Effekt zu programmieren. Als Unterstützung wird uns aber trotzdem das Programm »Raster-Utility« von Thomas Schlottke begleiten – zur Erleichterung. Sollten Sie nach der Lektüre dieses kleinen Kurses vorhaben, selbst Ra-IRQs zu programmieren, so benötigen Sie dazu allerdings Maschinensprache-Kenntnisse. Denn Basic ist nicht nur zu unflexibel dazu, sondern vor allem viel zu langsam.

Beginnen Sie am besten mit der Eingabe des Programmes »Raster-Utility« (Listing 1). Verwenden Sie dazu den MSE, Hinweise finden Sie auf Seite 159. Das kleine Demoprogramm (Listing 2) ist in Basic geschrieben und zeigt, sozusagen als Appetitanreger, einige der Möglichkeiten, die Listing 1 eröffnet.

Wenn Sie sich speziell dafür interessieren, was Raster-Interrupts sind, wie sie zustandekommen und wie sie in Assembler (ohne das »Raster-Utility«) programmiert werden, lesen Sie den Testkasten 1.

Laden Sie das Programm »Raster-Utility« jetzt mit dem Befehl

```
LOAD "RASTER-UTILITY",8,1 <RETURN>
```

Danach müssen Sie den Befehl NEW eingeben, damit alle Zeiger richtig gestellt werden. Laden Sie jetzt das Demoprogramm (Listing 2) wie ein normales Basic-Programm und starten Sie es mit

```
RUN <RETURN>
```

Der erste Teil des Beispielprogramms demonstriert, wie mit Hilfe des Raster-Interrupts gleichzeitig Text und Grafik verwendet werden kann: In der Mitte des Schirms wird ein Grafik-Fenster eingeblendet. Das anschließende Löschen der Grafik dauert in Basic einige Sekunden, gedulden Sie sich also. Danach werden einige Wellenlinien in das Fenster gezeichnet.

Kurzinfo: Rasterinterrupt

Programmart: Tool zur Rasterinterrupt-Programmierung
Laden: LOAD "Raster-Utility",8,1
Start: Nach dem Laden NEW und SYS 49152 eingeben
Besonderheiten: Das Demo-Programm lädt den Maschinenspracheteil nach und startet ihn mit SYS 49152. Das Programm belegt keinen Basic-Speicherplatz. Den dokumentierten Quelltext bitte nicht eingeben.
Programmautor: Thomas Schlottke

Wenn Sie jetzt eine Taste drücken, gelangen Sie in Teil 2 des Demoprogramms. Hier zeigt der C 64, daß er durchaus mehrere Farben gleichzeitig als Bildschirmhintergrund und -rahmen darstellen kann. Nach einem weiteren Tastendruck wird, passend zum 40. Geburtstag der BRD, die deutsche Fahne eingeblendet. Auch sie wird durch geschicktes Umschalten der Farben mittels Ra-IRQ erzeugt.

Das letzte Beispiel, das dann in einer Endlosschleife läuft, zeigt, daß verschiedene Zeichensätze zur selben Zeit verwendet werden können. Die Grenze kann sogar mitten durch eine Zeile gelegt werden oder, wie es das Demo beweist, wandern.

Nach dieser Spielerei wollen wir jetzt untersuchen, wie Sie mit dem »Raster-Utility« eigene Raster-IRQs programmieren.

Nach dem Laden des Maschinenprogramms können Sie mit der Programmierung beginnen. Zunächst sollten Sie sich überlegen, ob das Raster-Utility als Basic-Erweiterung oder als unabhängiges Maschinenprogramm benutzt werden soll. Beides ist möglich. Gegen eine Einbindung als Basic-Befehlserweiterung spricht eigentlich nur, daß dazu die Basic-Vektoren verbogen werden müssen. Andere Befehlserweiterungen sind damit ausgeschaltet. Ist das nicht

Vier neue Befehle

erwünscht, benutzen Sie das Programm als eigenständiges Assemblerprogramm, das etwas umständlicher über SYS-Befehle gesteuert wird. Im Einzelfall sollte das jeder selbst entscheiden, im Zweifelsfall hilft Probieren. Doch um das »Raster-Utility« auszuprobieren, können Sie es getrost als Erweiterung benutzen. Starten Sie es deshalb mit dem Befehl

```
SYS 49152
```

Das Utility ist jetzt mit seinen neuen Befehlen initialisiert. Diese bestehen aus zwei Zeichen: Das erste davon ist immer ein Punkt, auf den ein Buchstabe folgt. Vier neue Befehle gibt es (weitere Angaben, die eventuell entfallen können, stehen in geschweiften Klammern, die Klammer und die Ziffer gehören nicht zum Befehl):

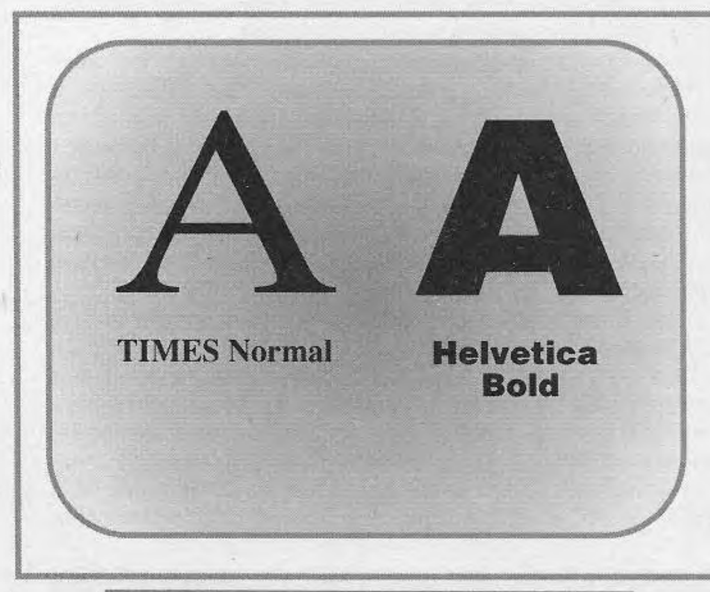
1) .S Rasterzeile, Register1, Inhalt1 {, Register2, Inhalt2{, ...}}

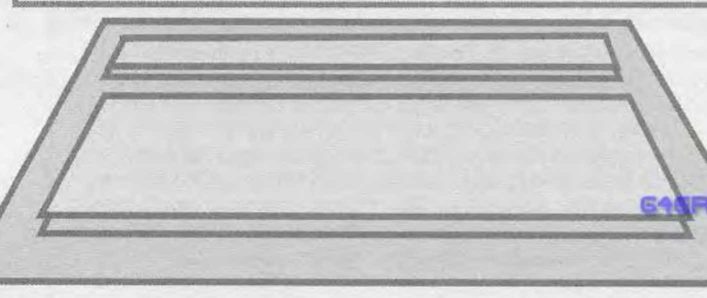
Der Set-Befehl ist wohl der wichtigste, da er festlegt, bei welchen Rasterzeilen welche Register wie verändert werden sollen. Hinter dem »Befehlswort« ».S« geben Sie zunächst die Rasterzeile an, die hier definiert werden soll. Der erlaubte Bereich liegt zwischen 1 und 255 einschließlich. Dahinter geben Sie, durch Komma getrennt, die Speicherzelle an, die in dieser Zeile geändert werden soll. Üblicherweise werden es VIC-Register, also Speicherzellen von 53248 bis 53295, sein. Hinter dieser Angabe folgt, wieder durch ein Komma getrennt, der gewünschte neue Inhalt für diese Speicherzelle. Sollen in dieser Zeile weitere Register beeinflußt werden, geben Sie die entsprechenden Daten, durch weitere Kommata getrennt, einfach dahinter an. Die Anzahl ist nur durch die Länge einer Basic-Zeile und den Speicher begrenzt. Beispiel:

```
.s 150,53280,0
```

Ab Rasterzeile 150 (liegt etwa in der Mitte des Bildschirms) soll Speicherzelle 53280 auf Null gestellt werden, der Bildschirmrahmen soll also schwarz werden. Dieser

nicht nur für Profis





Mit Raster-Interrupts lassen sich bequem Text und Grafik oder zwei verschiedene Zeichensätze gleichzeitig auf dem Bildschirm darstellen

Befehl alleine bewirkt lediglich, daß der gesamte Rahmen schwarz wird. Denn nachdem er in Rasterzeile 150 verfärbt wurde, gibt es keinen weiteren Befehl mehr, der die alte Farbe wiederherstellt. Hier müßten Sie, wie es weiter unten noch gezeigt wird, mit einem weiteren .S-Befehl arbeiten.

2) .D Rasterzeile

Durch den DELete-Befehl wird ein mit dem SET-Befehl programmierter Raster-Interrupt gelöscht. Hinter diesem Befehl geben Sie nur die Nummer der Rasterzeile an, in der nichts mehr geschehen soll.

Achtung!

Durch diesen Befehl werden auch alle Definitionen gelöscht, die nach dem betreffenden SET-Befehl eingegeben wurden. Der etwas fehlerhafte DEL-Befehl sollte daher also besser nicht so oft verwendet werden. Beispiel:

```
.d 150
```

In Zeile 150 (siehe Beispiel oben) geschieht nichts mehr.

3) .C

Durch den Clear-Befehl werden alle Raster-Interrupts abgeschaltet, aber nicht aus dem Speicher gelöscht. Der Bildschirm ist also wieder im Normalzustand. Dieser Befehl benötigt keine Parameter. Wiedereinschalten geschieht durch den Befehl

4) .I

Der Gegenbefehl zu ».C«, der Init-Befehl, läßt alle Raster-Interrupts zu, ist somit natürlich nur nach einem Clear-Befehl sinnvoll, da er bei SYS 49152 automatisch ausgeführt wird. Auch hier sind keine Parameter erforderlich.

Das Arbeiten mit dem Raster-Utility erklären wir am besten an einem Beispiel: Sie möchten den Bildschirmrand und den Textbildschirm zweifarbig darstellen. Also lösen Sie je einen Interrupt in der ersten Rasterzeile (ganz oben im Bildschirmrahmen) aus und einen weiteren in der Mitte des Bildschirms, etwa bei Rasterzeile 150. Sobald der Elektronenstrahl in der ersten Zeile ist, soll der Bildschirm schwarz gefärbt werden. Ab der 150. Rasterzeile ist wieder gelb erwünscht. Ein entsprechendes Programm sähe folgendermaßen aus:

```
10 SYS 49152                 : REM neue Basic-Befehle ein
20 .C                        : REM zur Sicherheit
30 .S   1,53281,0,53280,0    : REM Rahmen und Hintergrund
                               in Zeile 1 schwarz
40 .S 150,53281,7,53280,7    : REM ab Zeile 150 wieder gelb
50 .I                        : REM Effekt einschalten
```

Vor dem Start mit RUN muß natürlich Listing 1 im Speicher stehen. Wenn das kleine Programm eingegeben und gestartet wurde, sehen Sie den zweifarbigen Schirm. Geben Sie jetzt mal im Direktmodus (ohne Basic-Zeilennummer) ein:

```
.s 100,53280,2
```

Der Rahmen soll also zwischen Zeile 100 und 150 rot gefärbt werden. Die deutsche Fahne ist zu sehen. Nachdem der Rand sich geändert hat, probieren Sie einmal, die letzte Veränderung in Zeile 150 wieder zu löschen:

```
.d 100
```

Hätten Sie nach dem letzten SET-Befehl weitere SET-Befehle eingegeben, wären diese ebenfalls gelöscht worden (Fehler im DEL-Befehl). Zum Schluß kann man durch abwechselnde Eingabe von .C und .I diese beiden Befehle noch ausprobieren.

Zur Auswahl stehen alle Rasterzeilen von 1 bis 255. Derjenige, der sich schon etwas damit auskennt, wird sich fragen, warum nur 255 Rasterzeilen (anstatt 280) genutzt wurden. Die Antwort liegt auf der Hand. Löst man etwa in Zeile 0 einen Interrupt aus (mit »Raster-Utility« nicht möglich), so liegt dieser noch außerhalb des oberen Bildschirmrandes, während die Rasterzeile 256 zwar noch im sichtbaren Bereich liegt, jedoch weit unter dem Textbildschirm im Rahmen. Dieses ist zwar fernseh- beziehungsweise monitorspezifisch verschieden, jedoch erschien diese Einschrän-

Einschränkungen

kung aus Gründen der einfacheren Programmierung sinnvoll, da zwei Nullbytes im Datenfeld als Endkennzeichnung für eine Rasterzeile dienen und eine Rasterzeile über 255 nur in 16-Bit-Darstellung zu verarbeiten wäre.

Die Anzahl der Rasterzeilen-Interrupts, die Sie gleichzeitig anwenden können, ist ebenfalls begrenzt. Sie hängt vor allem vom Abstand dieser zueinander ab, je ein Raster-Interrupt in Zeile 150 und ein weiterer in Zeile 153 ist leider nicht möglich, da hier das Maschinenprogramm nicht mehr mitkommt. Der Grund für diese Beschränkung liegt in der hohen Geschwindigkeit des Elektronenstrahls, der den Bildschirm aufbaut (siehe Textkasten 1).

Beim SET-Befehl darf als Speicherzelle, die verändert werden soll, nicht die Null gewählt werden, was jedoch auch kaum sinnvoll wäre. Eine Prüfung erfolgt allerdings

nicht! Auch die Null als Nummer der Rasterzeile sollten Sie nicht verwenden. Achten Sie auch darauf, daß niemals zwei verschiedene SET-Befehle für dieselbe Rasterzeile verwendet werden. Der zweite Befehl würde nicht berücksichtigt:

Falsch:

```
.s 123,53280,0
.s 123,53281,6
```

Richtig:

```
.s 123,53280,0,53281,6
```

Die Tabelle, in der die SET-Befehle gespeichert werden, umfaßt nur 255 Byte. Läuft sie über, weil zu viele Definitionen erfolgten, arbeitet das Programm nicht mehr richtig. Daher sollte die Anzahl der SET-Befehle nicht übermäßig groß werden. Jeder SET-Befehl belegt in dieser Tabelle 3 Byte, dazu kommen noch je 3 Byte für jedes Register, das verändert werden soll. Die maximale Anzahl von SET-Befehlen mit je einem Register beträgt demnach etwa 42. Das Programm testet nicht, ob der neue »Job« noch in der Tabelle Platz hat. Übrigens beginnen die Interrupts mit steigender Zahl von SET-Befehlen sehr stark zu flimmern. Beachten Sie in diesem Zusammenhang bitte auch, daß mit dem DEL-Befehl gelöschte Rasterdaten immer noch Speicherplatz in der Tabelle belegen. Da dieser Befehl, wie oben mehrfach erklärt, auch nicht immer ganz fehlerfrei arbeitet, sollten Sie ihn in der Praxis vermeiden. Da der SET-Befehl nicht selbständig den Interrupt abschaltet, sollten Sie dies erledigen:

Vor jedem SET-Befehl sollte ein .C-Befehl stehen, danach ein .I-Befehl, um den Interrupt wieder einzuschalten (siehe Demoprogramm, Listing 2, Zeilen 200 bis 240). Ansonsten kann das Programm, insbesondere, wenn sehr viele Ra-IRQs programmiert sind, abstürzen.

Oben wurde bereits erwähnt, daß man anstelle der neuen Basic-Befehle auch eine Steuerung über SYS-Befehle vornehmen kann. Diese Möglichkeit werden wir Ihnen gleich noch genauer vorstellen. Hier sei nur der Hinweis angebracht, daß Sie zwar auch nach Benutzung der Steuerung über SYS-Befehle mit SYS 49152 die Befehlserweiterung einschalten dürfen, jedoch nicht umgekehrt. Sind erst einmal die neuen Befehle eingebaut, ist die Steuerung mit SYS »tabu«.

Diese Einschränkungen sind durch die Programmierung des Raster-Utility gegeben, wiegen in der Praxis jedoch nicht allzu schwer. Außerdem ist dieses Programm ja nur als Anregung zu verstehen, mit Hilfe von Listing 3 können Sie nach Belieben zusätzliche Prüfungen und Verbesserungen einbauen. Drei weitere Einschränkungen sind C64-spezifisch: Steht einer der neuen Befehle hinter dem THEN-Befehl, ist dazwischen ein Doppelpunkt notwendig:

Was sind Rasterinterrupts?

Um diesen Begriff zu erklären, muß man zunächst auf die beiden Wörter eingehen, die darin enthalten sind: »Raster« und »Interrupt«.

In jedem Fernseher und Monitor ist eine Art Kanone enthalten, die von hinten auf die Mattscheibe Elektronen schießt, welche dort die Mattscheibe zum Leuchten bringen. Eine Elektronik steuert diesen Strahl, den sogenannten »Rasterstrahl« nun so, daß er das Bild, das zu sehen sein soll, aufbaut. Der Strahl »tastet« die Scheibe zeilenweise ab, dabei werden genau die Punkte gesetzt, die der Fernsehsender oder in unserem Fall der Rechner vorschreibt. Damit der Computer den Strahl also in der Helligkeit verändern und so das Bild erzeugen kann, muß er wissen, welche Zeile des Monitorbildes vom Strahl gerade aufgebaut wird. In Europa besteht die sogenannte »PAL«-Norm (**P**hase **A**lternation **L**ine), die unter anderem vorschreibt, daß ein Fernsehbild etwa 650 solcher Zeilen besitzt. 50mal in der Sekunde überstreicht der Rasterstrahl das Bild (50 Hz Bildwiederholfrequenz), die Nummer der Zeile, die gerade aufgebaut wird, wechselt also etwa 32500mal in der Sekunde. Beim C64 sind es nicht ganz so viele Zeilen, hier besteht ein Bild aus exakt 280 Rasterzeilen. Wenn Sie sich das Bild Ihres Computers genau ansehen, werden Sie feststellen, daß alle Buchstaben aus waagerechten Strichen bestehen (gut zu erkennen beim Cursor). Dies sind die Rasterzeilen (siehe hierzu auch Textkasten 3). Da der Videocontroller (VIC) im C64 die Steuerung des Strahls übernimmt, muß ihm zu jeder Zeit bekannt sein, welche Zeile gerade angepeilt wird. Diese Angabe findet sich im Register 18 ($12) des VIC, es hat die Adresse 53266. Da sich der Inhalt 14000mal in der Sekunde ändert, kann man dieses Register übrigens gut als Zufallsgenerator verwenden. Das aber nur nebenbei.

Nun zum Interrupt: Ein beliebtes Beispiel zur Erklärung ist das folgende: Stellen Sie sich vor, während Sie gerade diesen Text lesen, klingelt das Telefon. Sie legen das Heft beiseite, um zu telefonieren. Danach nehmen Sie die Lektüre wieder auf, um weiterzulesen. Genau das passiert auch im C64: Er bearbeitet das Hauptprogramm, dies ist seine »Lektüre«. Jetzt wird ein Interrupt ausgelöst, der Computer unterbricht die Bearbeitung, erledigt einige Aufgaben wie das Cursorblinken, die Abfrage der Tastatur und das Weiterzählen der Uhren TI und TI$, danach wird das Hauptprogramm weiter bearbeitet – bis zum nächsten Interrupt. Diese Art von Interrupt nennt man IRQ (Interrupt **R**e**Q**uest), er wird von einer Uhr (einem »Timer«) 60mal in der Sekunde ausgelöst.

Koppelt man nun die Begriffe »Raster« und »Interrupt«, kommt man auf den »Rasterinterrupt«. Denn auch der VIC ist in der Lage, einen Interrupt auszulösen, beispielsweise, wenn der Lichtgriffel einen Punkt registriert – oder wenn der Rasterstrahl im Monitor eine bestimmte Zeile erreicht hat! Um sicherzustellen, daß der VIC einen IRQ auslöst, setzt man das niedrigste und das höchste Bit (dezimal also 129) im IMR-Register (Nummer 26, Adresse 53274) des VIC (siehe $c03a in Listing 3). Nun müssen wir ihm noch mitteilen, bei welcher Zeile der IRQ erfolgen soll. Dies geschieht durch Schreiben der Nummer in Register Nummer 18 (53266). Da dieses Register nur Zahlen von 0 bis 255 aufnehmen kann, es aber bis zu 280 Zeilen geben kann, wird noch ein achtes Bit benötigt, das wir in Adresse 53265 (Register Nr. 17) finden: Es ist dort das Bit 7 mit der Wertigkeit 128. Um etwa bei Zeile 123 einen IRQ auslösen zu lassen, löschen wir Bit 7 in 53265 (123 < 256) und schreiben 123 in Adresse 53266. Das ist auch schon alles.

Doch wie reagieren wir nun, wenn wir einen IRQ feststellen? Wurde dieser, wie üblich, durch den Timer ausgelöst (sogenannter »System-IRQ«), oder war es ein Rasterinterrupt? Dazu programmieren wir eine neue IRQ-Routine, indem wir den Zeiger $314/5 verbiegen (siehe einschlägige Literatur). Jetzt testen wir das IRR (siehe Tabelle, Adresse 53273): Sind wieder die Bits 0 und 7 gesetzt, war es der VIC (und damit der Rasterstrahl), der den IRQ auslöste, sonst der Timer. Im letzteren Falle können wir getrost an die Original-IRQ-Routine bei $ea31 verzweigen: Der Rasterstrahl ist noch nicht so weit.

Bei einem Raster-IRQ können nun zum Beispiel die VIC-Register verändert werden. Das altbekannte Beispiel ist der zweifarbige Bildschirmrahmen: Oben soll der Schirm von einem blauen Rahmen umgeben sein, unten soll es ein roter sein. Lassen wir also in der Mitte des Bildschirms (z. B. Zeile 150) einen Rasterinterrupt auslösen. Wird er registriert, soll der Bildschirmrahmen (Farbregister 53280) auf rot (Wert 2) gesetzt werden. Das reicht aber noch nicht: Tun wir nur dies, gelangt der Strahl irgendwann einmal zu Zeile 150, löst den IRQ aus, der Rahmen wird rot gesetzt – fertig. Er bleibt rot. Wir müssen dafür sorgen, daß außerhalb des sichtbaren Bereichs, etwa in Rasterzeile eins, wieder auf blau umgeschaltet wird – wieder mit einem Rasterinterrupt (siehe Bild).

Da, wie gesagt, 50mal in der Sekunde 300 Zeilen aufgebaut werden, und somit der Rasterstrahl 14000mal pro Sekunde die Zeile wechselt, ist Basic eindeutig zu langsam für Rasterinterrupts. Auch fehlen hier die Möglichkeiten der Interrupt-Programmierung. In Maschinensprache können aber mit relativ einfachen Programmen Rasterinterrupts programmiert werden.

Hier gibt es auch noch ein Betätigungsfeld für »Vollblutprogrammierer« und Profis: Wer schafft es, eine Hardcopy-Routine zu schreiben, die Raster-IRQs aller Art berücksichtigt? Beispielsweise »Uniprint« aus 64'er, 4/88 und Sonderheft 32 könnte man derartig erweitern. Wenn Sie wissen, wie so etwas funktionieren könnte, schreiben Sie ein Programm und schicken Sie es ein. Eine Veröffentlichung wäre, wenn es tadellos arbeitet und jeden VIC-Interrupt erkennt, fast sicher.

(Nikolaus Heusler/ag)



Weg mit dem Bildschirmrahmen

Oben wurde bereits angesprochen, daß man mit Hilfe der Rasterprogrammierung Sprites im oberen und unteren Bildschirmrahmen unterbringen kann. Da dies ein Spezialkapitel ist, wurde hierfür ein eigener Exkurs vorgesehen.

Geben Sie Listing 4 bitte mit dem MSE ein (Seite 159) und speichern Sie es. Als Beispiel können Sie Listing 6 verwenden, das Listing 4 automatisch nachlädt.

Der »Expander« wird mit

```
LOAD "BORDERLESS",8,8
NEW
SYS 51000
```

geladen und gestartet. Jetzt können Sie im oberen und unteren Rahmen (seitlich geht's leider nicht so einfach – in den Literaturhinweisen, Nummer 7 (64'er, 1/88, Seite 62) finden Sie speziell dazu Tips) Sprites unterbringen – ohne besondere Kniffe. Setzen Sie einfach die Y-Koordinate der Sprites auf die entsprechenden Werte.

Die Erklärung, wie dieser Effekt möglich wird, ist etwas kompliziert. Sogar die Experten sind sich uneinig darüber, was genau der Grund dafür ist: Ein Fehler des Videobausteins, eine bisher nicht bekannte Betriebsart, ein Monitorfehler oder was auch immer.

In Zeile 51 (obere Grenze zum Rahmen) und 250 (untere Grenze, diese beiden Werte sind der Schlüssel zum Erfolg) wird ein Rasterinterrupt ausgelöst. Wie Sie wissen, kann durch die Bits 0 bis 3 des Kontrollregisters 53265 die Y-Position des Bildschirms in acht Stufen eingestellt und die Größe (24/25 Zeilen) umgeschaltet werden. Schaltet man nun am unteren Bildschirmrand auf 25 Zeilen und verschiebt das Fenster nach oben, so »vergißt« der VIC, daß er sich ja mittlerweile im Rahmen befindet, und zeichnet den normalen Hintergrund (Farbe 53281) weiter. Dadurch, daß jetzt kein Rahmen mehr dargestellt wird, werden die Sprites sichtbar. Nun muß noch am oberen Rand (Zeile 51) wieder auf 25 Zeilen zurückgeschaltet werden, und der gewöhnliche Bildschirm wird nicht beeinflußt.

Der Effekt wird deutlich, wenn Sie nach dem Start etwa eingeben:

```
POKE 53281,11 : POKE 53280,0
```

Der Rahmen über und unter dem Textbildschirm fehlt einfach!

Das »Raster-Utility« ist für Spielereien dieser Art leider nicht geeignet, da es aufgrund seiner flexibleren Programmierung zu langsam ist und nicht rechtzeitig und genau gleichzeitig umschaltet.

Listing 5 dokumentiert die Arbeitsweise des Programms »Borderless«. Nur der Befehl ab $c743 bedarf einer näheren Erklärung. Geben Sie zur Probe nach dem Start von »Borderless« einmal ein:

```
POKE 16383, 8
```

Am oberen und unteren Bildschirmrand erscheinen senkrechte Striche. Der VIC weiß nicht, was er in dem neu gewonnenen Bereich darstellen soll (Text funktioniert leider nicht), und so nimmt er das Hires-Byte aus der letzten Speicherzelle seiner Bank, in diesem Fall $3fff (16383). Vorsicht ist geboten, da diese Zelle mitten im Basic-Speicher liegt. Durch diese Eigenart kommen auch die »Wanderungen« der Linien in Listing 6 zustande.

Damit keine unnötigen Verzögerungen auftreten, schaltet »Borderless« den Timer-CIA-System-IRQ kurzerhand ab. Dafür wird am Ende des Ra-IRQs der normale Interrupt ausgeführt. Da die Raster-IRQs etwas schneller kommen als 60mal in der Sekunde, blinkt der Cursor jetzt etwas schneller, und die Cursortasten reagieren schneller.

Kopiert man aus dem Zeichensatz-ROM Zeichen in Sprites, können Sie über diesen Umweg sogar Texte im Rahmen darstellen. Zum Beispiel das Zeichenprogramm »EGA« arbeitet nach diesem Prinzip.

(Nikolaus Heusler/ag)

Falsch:

```
IF OFF = 1 THEN .C
```

Richtig:

```
IF OFF = 1 THEN : .C
```

oder:

```
IF OFF = 1 THEN LET OFF = 1 : .C
(denn LET ist kein neuer Befehl)
```

Da der normale System-Interrupt nicht mehr ganz so regelmäßig abläuft, gehen als Folge die Uhren TI und TI$ falsch. Außerdem sollten Sie Operationen mit der Floppy, der Datasette oder einem Drucker vermeiden, während ein Rasterinterrupt läuft. Abstürze sind sonst nicht auszuschließen, da diese Operationen einen ganz genauen und sauberen IRQ benötigen.

Steuerung über SYS-Befehle

Nun soll, wie versprochen, die alternative Steuerung über SYS-Befehle erklärt werden. Soll das »Raster-Utility« zusammen mit einer anderen Erweiterung betrieben werden, können die vier neuen Befehle nicht installiert werden. In diesem Fall betreiben Sie Listing 1 über SYS-Befehle:

```
Ein    = 49172 (Utility einschalten)
Init   = 49190
Set    = 49275
Clear  = 49504
Delete= 49524
```

Der Befehl SYS 49152 darf in diesem Fall **nicht mehr** verwendet werden. Um in dieser Betriebsart ein Textfenster zwischen Rasterzeile 100 und 200 in eine Hires-Grafik ab $2000 einzublenden, dient etwa folgendes Programm:

```
10 SYS 49172                       :REM alles einschalten
20 SYS 49504                       :REM Clear
30 SYS 49275,100,53272,21,53265,27:REM ab Zeile 100
                                   Textmodus
40 SYS 49275,200,53272,29,53265,59:REM ab Zeile 200
                                   wieder Grafik
50 SYS 49190                       :REM Init
```

Ein ähnliches Beispiel finden Sie im Demoprogramm (Listing 2). Doch nun wird Sie interessieren, wie denn das »Raster-Utility« intern arbeitet und funktioniert.

Das »Raster-Utility« ist ein vollkommen in Assembler geschriebenes Programm, das im Speicher von $c000 bis $c198 liegt (siehe Listing 3). Der Bereich ab $c199 wird für das Datenfeld verwendet. Ein Datensatz für eine Rasterzeile baut sich folgendermaßen auf:

```
  1.Byte: Rasterzeile (oder Nullbyte, wenn Tabellenende)
2-3.Byte: Register in 16-Bit-Darstellung (Low-/High-Byte)
  4.Byte: neuer Registerinhalt
5-6.Byte: ggf. Register Nr. 2 wie oben
  7.Byte: ggf. Inhalt für Register Nr. 2
und so weiter für alle Register, die zu dieser
Rasterzeile gehören
danach: zwei Nullbytes als Endkennzeichnung für eine
Rasterzeile
```

Danach folgt der Datensatz für die nächste Rasterzeile.

Die Datensätze stehen in der Reihenfolge im Speicher, in der sie mit den SET-Befehlen festgelegt wurden. Die Größe des Gesamtfeldes ist von der Anzahl der Rasterzeilen und der Anzahl der zu verändernden Register abhängig.

Die IRQ-Vektoren $0314/5 werden auf die neue Interrupt-Service-Routine bei $c0b6 verbogen. Bindet man das Raster-Utility in eine Basic-Erweiterung ein, werden die Vekto-

Raster- und Bildschirmzeilen

Sie kennen jetzt zwei Begriffe: Rasterzeile und Bildschirmzeile. Der normale Textbildschirm besteht aus 25 Zeilen. Jede dieser sogenannten »Textzeilen« besteht aus genau acht Rasterzeilen (vgl. Zeichensatzeditoren: Ein Zeichen ist acht Pixel hoch). Das gesamte Textfenster umfaßt also exakt 8 mal 25 = 200 Rasterzeilen. Es gilt folgende Aufteilung:

```
Rasterzeile   0 bis  50: oberer Bildschirmrahmen
Rasterzeile  51 bis 250: normaler Text-/Grafikbildschirm
Rasterzeile 251 bis 280: unterer Bildschirmrahmen
```

Wichtig ist auch die Formel zur Umrechnung einer Textzeile in die entsprechenden Rasterkoordinaten:

```
Rasterzeile = [Nummer der Textzeile (0 bis 24)] * 8 + 50
```

Beispiel: Die Textzeile 12 umfaßt die Rasterzeilen 146 bis 153 einschließlich.

(Nikolaus Heusler/ag)

ren zur Ausführung eines Basic-Befehls ($0308/9) auf eine neue Befehls-Routine bei $c043 verbogen.

Die genaue Funktionsweise des Programms ersieht man am besten aus einem kommentierten Assemblerlisting. Sehen Sie sich Listing 3 (nicht eingeben!) an: Hier wurde jeder Assemblerbefehl genau erklärt. Wenn Sie bis jetzt die Rasterzeilen nur aus der Literatur kannten, aber Ambitionen in dieser Richtung haben, sehen Sie sich das Listing einmal genauer an, es ist sehr lehrreich.

Wenn Sie nun das Programm an eine andere Adresse als $c000 assemblieren wollen, können Sie auch Listing 3 (Seite 110) verwenden. Vergessen Sie in diesem Fall aber nicht, neben den absoluten Adressen (JMP-Befehle und so weiter) auch die Pointer in $c000 (Vektor $306/7), $c01e (zeigt auf 255 Byte freien Speicherplatz, etwa bei $c199) sowie $c026 (Vektor $314/5) anzupassen. Auch der Vektor am Programmende ($c193) muß angepaßt werden, er soll auf einen RTS-Befehl im Programm oder (besser) im Kernel zeigen (z. B. bei $e194).

Was ist sonst noch möglich?

Wie Sie die neuen Erkenntnisse über die Rasterprogrammierung anwenden, liegt letztendlich in Ihrem Ermessen. Gerade das Mischen von Text und Grafik bietet viele Möglichkeiten, etwa ein Textadventure mit eingeblendetem Grafikfenster oder ein Kurvendiskussionsprogramm, das auf dem Grafikschirm die Funktion plottet und auf Wunsch ein Textfenster einblendet, in dem der Funktionsterm, weitere Erklärungen oder die Ergebnisse der Diskussion zu finden sind. In einem Zeichensatzeditor können die Titel- und Menüzeile im normalen Zeichensatz dargestellt werden, das Editorfeld im Grafikmodus und die komplette Anzeige des Zeichensatzes in einem dritten Fenster, in dem die neuen Zeichen aktiviert sind. Vor allem in Spielen werden Rasterinterrupts eingesetzt, um (viel) mehr als »nur« acht Sprites gleichzeitig auf dem Bildschirm darzustellen. Dazu kopieren Sie ab einer bestimmten Rasterzeile Werte aus dem RAM in die Spriteregister des VIC, zum Beispiel sind in der unteren Bildschirmhälfte dann die nächsten acht Sprites zu sehen. Beachten Sie bitte dazu auch das Programm »Provic 64« auf Seite 90. Auch Softscrolling wird erst durch Ra-IRQs möglich. Läßt man die Nummer der Zeile, in der eine Umschaltung geschieht, variabel, wie im Demoprogramm ab Zeile 350, so sind Effekte denkbar wie ein Grafikfenster,

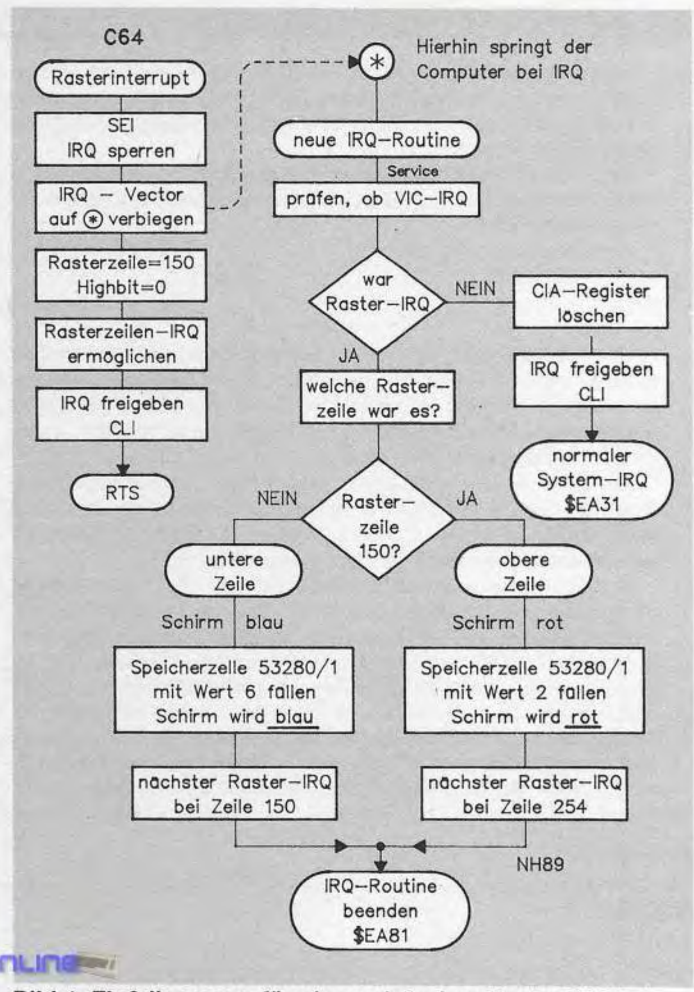


Bild 1. Flußdiagramm für einen einfachen Raster-IRQ

das sich langsam über einem Textbildschirm aufzieht, und vieles mehr. Ein sehr interessanter Effekt sind die Sprites, die Sie mit Hilfe des Ra-IRQ auch im oberen und unteren Bildschirmrahmen (!) plazieren können. Ein Beispiel dazu findet sich in Listing 4 (bitte mit dem MSE eingeben, Hinweise auf Seite 159, es ist in Listing 5 dokumentiert). Beachten Sie dazu auch den Textkasten zwei. Dieses Programm wird absolut geladen und nach NEW mit

```
SYS 51000
```

Raster-IRQ mit dem VIC

Register 18:	Beim Lesezugriff wird die Nummer der Rasterzeile, die gerade auf dem Schirm aufgebaut wird, ausgegeben. Bit 7 von Register 17 dient als Übertrag für Rasterzeilen über 255. Beim Schreibzugriff wird die Rasterzeile festgelegt, bei der ein IRQ ausgelöst werden soll.
Register 25:	Interrupt-Request-Register (IRR). Dieses Register signalisiert, daß der VIC einen IRQ ausgelöst hat. Die einzelnen Bits geben die Quelle an: Bit 0: Rasterzeile Bit 1: Kollision Sprite mit Hintergrund Bit 2: Kollision zweier Sprites Bit 3: Am Lightpen wurde ein Strobe ausgelöst. Die X/Y-Position finden Sie in den Registern 19 (X-Koordinate) und 20 (Y) Bit 7: wird zusammen mit einem der anderen Bits gesetzt. Durch Beschreiben dieses Registers mit dem gelesenen Wert wird es gelöscht (muß vor Verlassen der IRQ-Routine geschehen).
Register 26:	Interrupt-Mask-Register (IMR). Die Bedeutung der Bits ist dieselbe wie die in Register 25. Dieses Register dient zum Ein-/Ausschalten der vier IRQ-Quellen im VIC. Wenn beim Beschreiben dieses Registers das Bit 7 gesetzt ist, werden im internen Schalter-Register die im Datenbyte gesetzten Quellen-Bits gesetzt, ist Bit 7 gelöscht, löscht der VIC die IRQ-Möglichkeiten, deren Bits beim Beschreiben von Register 26 gesetzt sind. Zwei Beispiele: Beschreiben mit %10000001: Raster-IRQ einschalten, andere Quellen nicht beeinflussen. Beschreiben mit %00001010: Lichtgriffel- und Sprite/Hintergrund-IRQ abschalten, andere Quellen nicht beeinflussen. Lesen dieses Registers ist nicht möglich.
Register 30 und 31:	In diesen Registern wird vermerkt, welches Sprite mit dem Hintergrund (Reg. 30) oder einem anderen Sprite (Reg. 31) kollidiert ist. Diese Register müssen nach dem Lesen manuell gelöscht werden.

Tabelle 1. Die Register des VIC, die sich mit dem Raster-IRQ befassen



gestartet. Genauere Informationen zu diesem Thema finden Sie im zweiten Textkasten. In der Tabelle sind alle Register des VIC, die zur Raster-IRQ-Programmierung wichtig sind, zusammengefaßt. Das Bild zeigt wie eine ganz einfache Raster-IRQ-Routine aufgebaut ist: Der Bildschirm soll über Rasterzeile 150 rot sein und darunter blau.

Das war also unser kleiner Ausflug in die weite Welt der Rasterprogrammierung. Wie hat es Ihnen denn gefallen? Sie sollten sich, so noch nicht geschehen, einmal ein wenig damit beschäftigen. Es ist schon toll, wie mit wenig Programmierarbeit Effekte erzeugt werden können, die ein Einsteiger als unmöglich bezeichnen würde. Literatur zu diesem Thema gibt es ja mehr als genug, auch in der 64'er.

(Nikolaus Heusler/ag)

Literatur zum Rasterzeilen-IRQ (Auszug):

1. Die Interrupts des Videocontrollers (64'er 5/87, S. 47)
2. 27 Zeilen auf dem Bildschirm (64'er 3/87, S. 84)
3. Erklärung zum vibrierenden Bildschirm (64'er 2/87, S. 22)
4. Flackern beim Rasterzeilen-IRQ (64'er 2/87, S. 79)
5. Soft-Scrolling auf dem C64 (64'er Sonderheft 4/85, S. 110)
6. Der Unterwasser-Effekt (64'er 4/88, S. 59)
7. Hyperscreen III - Sprites ohne Grenzen (64'er 1/88, S. 62)
8. Grafik im Bildschirmrahmen (64'er 11/88, S. 57)
9. Wellenbad auf dem Bildschirm (64'er 1/88, S. 59)
10. 112 Sprites (64'er 10/88, S. 47)
11. Der 64'er Sternenhimmel (64'er 9/88, S. 48)
12. Kopfzeilen per Raster-IRQ (64'er 4/88, S. 99)
13. (Nachtrag zu) $3fff (64'er 4/88, S. 99 und 1/88, S. 72)
14. Super-Rasterzeilen-IRQ (64'er 11/88, S. 70)
15. Assembler ist keine Alchimie (Teil 12) (64'er 9/85, S. 109)
16. Hires-3 (Teil 3) (64'er 8/85, S. 152)
17. Das Maschinensprachebuch für Fortgeschrittene, Lothar Englisch, Data Becker Verlag, S. 110


```
Name : raster-utility    c000 c19a
-----------------------------------
c000 : a2 43 a0 c0 8e 08 03 8c  d2
c008 : 09 03 a2 ae a0 a7 8e 93  ba
c010 : c1 8c 94 c1 a9 00 a2 14  c2
c018 : 9d 94 c1 ca d0 fa a2 99  6c
c020 : a0 c1 86 9e 84 9f a9 b6  70
c028 : 8d 14 03 a9 c0 8d 15 03  88
c030 : a9 00 ad 11 d0 29 7f 8d  d6
c038 : 11 d0 a9 81 8d 1a d0 8d  54
c040 : 12 d0 60 20 73 00 c9 2e  91
c048 : f0 06 20 79 00 4c e7 a7  c4
c050 : 20 73 00 c9 53 d0 03 4c  c3
c058 : 7b c0 c9 44 d0 03 4c 74  6d
c060 : c1 c9 43 d0 06 20 73 00  20
c068 : 4c 60 c1 c9 49 d0 09 20  0d
c070 : 26 c0 20 73 00 6c 93 c1  a2
c078 : 4c 08 af 20 9b b7 ac 95  0e
c080 : c1 8a 91 9e 20 fd ae 20  ab
c088 : eb b7 ac 95 c1 c8 a5 14  4e
c090 : 91 9e c8 a5 15 91 9e c8  41
c098 : 8a 91 9e 8c 95 c1 20 79  ff
c0a0 : 00 c9 2c f0 df c8 a9 00  99
c0a8 : 91 9e c8 91 9e c8 91 9e  a0
c0b0 : 8c 95 c1 6c 93 c1 ad 19  35
c0b8 : d0 8d 19 d0 30 07 ad 0d  bb
c0c0 : dc 58 4c 31 ea ad 12 d0  08
c0c8 : 8d 96 c1 a0 00 20 0f c1  e5
c0d0 : f0 27 cd 96 c1 f0 06 20  96
c0d8 : 13 c1 4c d0 c0 c8 b1 9e  4f
c0e0 : 85 f7 c8 b1 9e 85 f8 c8  55
c0e8 : b1 9e 48 98 aa a0 00 68  8e
c0f0 : 91 f7 8a a8 c8 b1 9e d0  6b
c0f8 : e7 ad 96 c1 20 32 c1 cd  ca
c100 : 96 c1 d0 05 a9 00 20 32  cb
c108 : c1 8d 12 d0 4c 81 ea b1  0e
c110 : 9e d0 1e c8 b1 9e d0 08  1a
c118 : c8 b1 9e d0 04 a9 00 60  c9
c120 : c8 c8 c8 b1 9e d0 f9 c8  9e
c128 : b1 9e d0 f5 c8 b1 9e f0  92
c130 : ec 60 8d 97 c1 a9 ff 8d  27
c138 : 98 c1 a0 00 20 0f c1 f0  3c
c140 : 14 cd 97 c1 90 0a f0 08  86
c148 : cd 98 c1 b0 03 8d 98 c1  6a
c150 : 20 13 c1 d0 ec ad 98 c1  a6
c158 : c9 ff d0 03 ad 97 c1 60  15
c160 : 78 a9 00 8d 1a d0 a2 31  74
c168 : a0 ea 8e 14 03 8c 15 03  92
c170 : 58 6c 93 c1 20 9b b7 8e  f6
c178 : 96 c1 a0 00 20 0f c1 f0  7a
c180 : 0a cd 96 c1 f0 06 20 13  34
c188 : c1 d0 f6 60 a9 00 91 9e  99
c190 : 6c 93 c1 5f c1 ff ff ff  3e
c198 : ff ff ff ff ff ff 00 00  97
```

Listing 1. Das Programm »Raster-Utility« erlaubt die einfache Programmierung von Raster-IRQs auch von Basic aus. Dieses Programm wird mit dem MSE (Seite 159) eingegeben.

```
0 POKE 53281,6:POKE 53280,6:PRINT"{CLR,DOW
  N,2RIGHT,WHITE}DEMOPROGRAMM 'RASTER-UTIL
  ITY'                                      <244>
2 FOR X=0 TO 18:PRINT:NEXT                  <199>
3 POKE 56,32:CLR:RUN 10                     <159>
4 PRINT"{BLACK,UP}DRUECKEN SIE BITTE EINE
  TASTE !{WHITE}                            <021>
5 POKE 198,.:WAIT 198,1:POKE 198,.          <042>
6 POKE 781,PEEK(214)-1:SYS 59903:RETURN     <134>
10 EIN   =49172                             <179>
20 INIT  =49190                             <143>
30 SET   =49275                             <077>
40 CLEAR =49504                             <062>
50 DEL   =49524                             <153>
60 :                                        <036>
70 SYS EIN                                  <071>
80 SYS SET ,152,53265,27,53272,21           <255>
90 SYS SET ,104,53272,29,53265,59           <158>
95 PRINT"{2UP}SO MISCHT MAN TEXT UND GRAFI
   K !{2UP}                                 <020>
100 FOR X=0 TO 1920:POKE 10432+X,0:NEXT     <171>
110 FOR Y=7 TO 12                           <066>
120 FOR X=0 TO 39                           <108>
130 POKE 4+Y*320+8192+X*8+4*SIN(X/2),36*(Y
    -INT(Y/2)*2)+219                        <147>
140 NEXT:NEXT                               <143>
160 GOSUB 4:PRINT"{DOWN}JETZT KOMMEN DIVER
    SE FARBSPIELCHEN...{2UP}                <062>
170 SYS CLEAR:POKE 53265,PEEK(53265)AND 22
    3:POKE 53272,PEEK(53272)AND 247         <246>
180 :                                       <156>
200 SYS 49152:.C                            <165>
210 FOR X=1 TO 8                            <048>
220 .S X*30,53281,X+2 ,53280,X+2:NEXT       <015>
230 .I:GOSUB 4:.C:SYS 49152                 <120>
240 :                                       <216>
300 .S 1,53280,0,53281,0                    <249>
310 .S 100,53280,2,53281,2                  <091>
320 .S 200,53280,7,53281,7                  <247>
325 GOSUB 4                                 <030>
330 SYS 58692:SYS 49152:PRINT"{HOME,4DOWN}
    WIE FINDEN SIE DAS:{2DOWN}              <255>
332 .S88,53272,21:.S136,53272,23            <149>
334 PRINT"OBEN HABEN WIR GROSS/GRAFIKZEICH
    EN:                                     <255>
336 PRINT"{DOWN,SPACE}ABCDEFGHIJKLMNOPQRS
    ABCDEFGHIKLMNOPQRS                      <171>
337 PRINT"{2DOWN}UND UNTEN KLEIN/GROSSBUCH
    STABEN:                                 <217>
338 PRINT"{DOWN,SPACE}ABCDEFGHIJKLMNOPQRS
    ABCDEFGHIKLMNOPQRS                      <173>
340 PRINT"{2DOWN}SEHR SCHOENE EFFEKTE SIND
     MACHBAR                                <154>
342 PRINT"{2DOWN,2SPACE}SIND DOCH SCHOENE
    SACHEN...                               <204>
344 .S240,53272,23:.S186,53272,21           <018>
350 FOR X=0 TO 14                           <078>
352 POKE 49579,X+178                        <140>
353 FOR J=.TO 30:NEXT                       <027>
354 NEXT                                    <110>
355 FOR X=13 TO 1 STEP-1                    <087>
356 POKE 49579,X+178                        <144>
357 FOR J=.TO 30:NEXT                       <031>
358 NEXT                                    <114>
360 GOTO 350                                <130>
```

Listing 2. Dieses Demoprogramm zeigt die Möglichkeiten des »Raster-Utility«

```
400 :                                         <122>
402 REM    DEMOPROGRAMM                       <155>
404 REM    'RASTER-UTILITY'                   <102>
406 REM    (C) MARKT & TECHNIK                <187>
408 REM    VON THOMAS SCHLOTTKE               <211>
410 REM    BEARBEITET VON N. HEUSLER          <051>
```

Listing 2. (Schluß)

```
                      ; Assemblerlisting zu >>
                        Raster-Utility<<
                      ; Programm von Thomas Schlottke
                      ; Kommentar von Nikolaus Heusler

                      ; Einsprung durch SYS 49152
                      ; Vektoren: IBAS und IRQ stellen
$c000 ldx   #$43      ; ab $c043 beginnt neue
                        Befehlsauswertung
$c002 ldy   #$c0      ; Highbyte
$c004 stx $0308       ; Basic-Befehl-ausführen Vektor auf
                        diese Adresse
$c007 sty $0309       ; >>verbiegen<<
$c00a ldx   #$ae      ; Pointer umstellen auf $a7ae
$c00c ldy   #$a7
$c00e stx $c193       ; Routinen umschalten auf Befehls-
                        Betrieb (siehe unten bei $c075)
$c011 sty $c194       ; und Highbyte
                      ; Varibalen löschen, Einsprung
                        >>Ein<<(Utility initialisieren)
                        mit SYS 49172
$c014 lda   #$00      ; Füllwert
$c016 ldx   #$14      ; 20 Speicherzellen löschen $c195
                        bis $c1a9
$c018 sta $c194,x     ; Zelle löschen
$c01b dex             ; noch eine Variable ?
$c01c bne $c018       ; ja
                      ; Zeiger auf Rasterzeilenspeicher
                        setzen
$c01e ldx   #$99      ; Speicher bei $c199
$c020 ldy   #$c1      ; Highbyte
$c022 stx   $9e       ; in Pointer $9e/9f
$c024 sty   $9f       ; und Highbyte
                      ; Befehl INIT (Raster-IRQ
                        einschalten, SYS 49190)
$c026 lda   #$b6      ; IRQ-Vektor ($0314/5)
$c028 sta $0314       ; auf $c0b6 setzen
$c02b lda   #$c0      ; Highbyte
$c02d sta $0315       ; eigentlich fehlen hier die
                        Befehle SEI und CLI
$c030 lda   #$00      ; blinder Befehl
$c032 lda $d011       ; Kontrollregister 1 des VIC
$c035 and   #$7f      ; Bit Nr. 7 (Wert 128) löschen
$c037 sta $d011       ; nur Rasterzeilen von 0 - 255
                        zulassen
$c03a lda   #$81      ; Interrupt (Bit 7) durch
                        Rasterzeile (Bit 0)
$c03c sta $d01a       ; erlauben (Interrupt Mask Register,
                        IMR)
$c03f sta $d012       ; erster Interrupt bei Zeile 129
                        (unnötig)
$c042 rts             ; fertig mit Initialisierung

                      ; neue Befehlsroutine
                        (Pointer $0308/9)
$c043 jsr $0073       ; nächstes Zeichen aus Basicprogramm
                        holen
$c046 cmp   #$2e      ; '.' ist es ein Punkt ?
$c048 beq $c050       ; ja, dann neuer Befehl
$c04a jsr $0079       ; sonst letztes Zeichen nochmal
                        holen, Flags setzen
$c04d jmp $a7e7       ; und normalen Basic-Befehl auswerten

                      ; neuer Befehl
$c050 jsr $0073       ; Zeichen nach Punkt holen
$c053 cmp   #$53      ; 'S' Befehl SET ?
$c055 bne $c05a       ; nein, weiter testen
$c057 jmp $c07b       ; sonst SET-Befehl behandeln
$c05a cmp   #$44      ; 'D' Befehl DELETE ?
$c05c bne $c061       ; nein, weiter testen
$c05e jmp $c174       ; sonst DELETE-Befehl bearbeiten
$c061 cmp   #$43      ; 'C' Befehl CLEAR ?
$c063 bne $c06b       ; nein, weiter testen

                      ; Befehl CLEAR (als neuer
                        Basic-Befehl, kein Einsprung)
$c065 jsr $0073       ; Zeichen nach '.C' holen
$c068 jmp $c160       ; und IRQ ausschalten

$c06b cmp   #$49      ; 'I' Befehl INIT ?
$c06d bne $c078       ; nein, dann falscher Befehl
$c06f jsr $c026       ; IRQ einschalten
$c072 jsr $0073       ; weiter in Basic
$c075 jmp($c193)      ; Befehl beendet
                      ; wurde das Maschinenprogramm nicht
                        über neue Basicbefehle, sondern
                        über SYS aktiviert, zeigt der
                        Vektor $c193/4 auf ein normales
                        RTS bei $c15f. Wurden mit SYS 49152
                        die neuen Basicbefehle aktiviert,
                        können die Routinen nicht über RTS
                        verlassen werden. Dann zeigt dieser
                        Pointer auf $a7ae.
$c078 jmp $af08       ; SYNTAX ERROR

                      ; Befehl SET (SYS 49275)
$c07b jsr $b79b       ; GETBYT, Komma o.ä. und Zahl hinter
                        Befehl (Rasterzeile) nach X holen
$c07e ldy $c195       ; Zeiger auf Ende der Rastertabelle
$c081 txa             ; Nummer der Rasterzeile
$c082 sta  ($9e),y    ; in Tabelle eintragen
                      ; keine Prüfung, ob der neue Auftrag
                        noch in der Tabelle Platz hat, oder
                        ob schon ein anderer Auftrag für
                        diese Rasterzeile gespeichert ist)
$c084 jsr $aefd       ; Komma holen
$c087 jsr $b7eb       ; GETADR, 16 Bit Zahl nach $14/15,
                        Komma und GETBYT, Wert nach X holen
$c08a ldy $c195       ; Zeiger auf Tabellenende
$c08d iny             ; Byte 1 enthält bereits die Nummer
$c08e lda  $14        ; Registernummer low
$c090 sta  ($9e),y    ; in Tabelle eintragen
$c092 iny             ; Position 3
$c093 lda  $15        ; und Highbyte
$c095 sta  ($9e),y    ; eintragen
                      ; kein Test, ob das Register die
                        Adresse Null hat (dann arbeitet
                        die Routine fehlerhaft)
$c097 iny             ; nächstes Register
$c098 txa             ; erwünschten Registerinhalt
$c099 sta  ($9e),y    ; ebenfalls eintragen
$c09b sty $c195       ; neuen Tabellenendezeiger vormerken
                      ; da die Adressierung der 255 Byte
                        langen Tabelle über das Y-Register
                        (8 Bit) erfolgt, läuft sie nach
                        einer bestimmten Anzahl von Jobs
                        über => Fehlfunktion
$c09e jsr $0079       ; Zeichen aus Programm holen
$c0a1 cmp   #$2c      ; ',' weitere Daten ?
$c0a3 beq $c084       ; ja, weiteres Register eintragen
$c0a5 iny             ; sonst als Endezeichen
$c0a6 lda   #$00      ; ein Nullbyte
$c0a8 sta  ($9e),y    ; eintragen (Registernummer low)
$c0aa iny             ; und noch eins
$c0ab sta  ($9e),y    ; (Registernummer high)
$c0ad iny             ; und ein letztes
$c0ae sta  ($9e),y    ; Tabellenendekennzeichen
$c0b0 sty $c195       ; Zeiger auf Tabellende
                        richtigstellen
$c0b3 jmp($c193)      ; fertig (siehe $c075)

                      ; neue IRQ-Routine
$c0b6 lda $d019       ; wurde IRQ durch Videochip
                        (Rasterzeile) ausgelöst ?
```

```
$c0b9 sta $d019         ; wenn ja, nächste Rasterzeile
                          erlauben (Register wurde durch
                          Lesen gelöscht)
$c0bc bmi $c0c5         ; war der Videochip
                        ; normaler IRQ
$c0be lda $dc0d         ; CIA Interrupt Steuerregister durch
                          Lesen löschen und nächsten Timer-
                          IRQ erlauben (verhindert Flimmern)
$c0c1 cli               ; Raster-IRQ während System-(Timer)-
                          IRQ erlauben, verhindert Flimmern
$c0c2 jmp $ea31         ; weiter im System-IRQ (alte Routine)

                        ; IRQ, ausgelöst durch Rasterzeile
$c0c5 lda $d012         ; Rasterzeilennummer, die IRQ
                          auslöste
$c0c8 sta $c196         ; merken
$c0cb ldy  #$00         ; ab der nullten Tabellenposition
                          suchen
$c0cd jsr $c10f         ; suchen, ob Rasterzeile, die IRQ
                          ausgelöst hat, vom Benutzer
                          definiert wurde
$c0d0 beq $c0f9         ; Tabellenende, dann fertig
$c0d2 cmp $c196         ; gefundene gleich gesuchter Zeile ?
$c0d5 beq $c0dd         ; ja, dann entsprechend verfahren
$c0d7 jsr $c113         ; sonst nächste Rasterzeile anwählen
$c0da jmp $c0d0         ; und auf Gültigkeit testen
                          (unsauber: hier fehlt Test,
                          ob Tabellenende)
                        ; Rasterzeile identifiziert
$c0dd iny               ; Zeiger um 1 erhöhen
$c0de lda  ($9e),y      ; Registeradresse low
$c0e0 sta  $f7          ; in Hilfszeiger
$c0e2 iny               ; nächsten Tabellenplatz anfahren
$c0e3 lda  ($9e),y      ; und Highbyte
$c0e5 sta  $f8          ; in Zeiger
                        ; $f7/f8 enthält jetzt Adresse des
                          Registers, das verändert werden
                          soll
$c0e7 iny               ; Zeiger auf gewünschten neuen Inhalt
$c0e8 lda  ($9e),y      ; neuen Inhalt aus Tabelle lesen
$c0ea pha               ; vorläufig merken
$c0eb tya               ; Zeiger (noch im Y-Register)
$c0ec tax               ; nach X bringen
$c0ed ldy  #$00         ; Y auf Null stellen (für $c0f0)
$c0ef pla               ; neuen Inhalt zurückholen
$c0f0 sta  ($f7),y      ; gewünschte Änderung ausführen
                          (Y = 0)
$c0f2 txa               ; Zeiger zurück nach Y
$c0f3 tay
$c0f4 iny               ; Zeiger auf nächstes Register
$c0f5 lda  ($9e),y      ; noch ein Datum ?
$c0f7 bne $c0e0         ; ja, dann ausführen
$c0f9 lda $c196         ; Nummer der Rasterzeile
$c0fc jsr $c132         ; sonst in Tabelle nächste zu
                          bearbeitende Rasterzeile suchen
$c0ff cmp $c196         ; höhere Rasterzeile gefunden ?
$c102 bne $c109         ; ja, dann ist diese als nächste dran
                        ; von vorn (oben am Bildschirm)
                          anfangen
$c104 lda  #$00         ; kleinste (erste) Zeile in Tabelle
                          suchen
$c106 jsr $c132         ; Suchroutine
$c109 sta $d012         ; als nächste Rasterzeile, die einen
                          IRQ auslösen soll, festlegen
$c10c jmp $ea81         ; Interrupt-Routine beenden

                        ; sucht ab Tabellenposition Y nach
                          nächstem nicht gelöschtem
                          Rasterzeilen-Datensatz. In A
                          erscheint die Nummer der neuen
                          Rasterzeile (oder Null für
                          Tabellenende) und in Y die
                          Tabellenposition
$c10f lda  ($9e),y      ; prüfen, ob Rasterzeile gelöscht
                          (erstes Datensatz-Byte lesen)
$c111 bne $c131         ; kein Nullbyte, also definiert
                          (RTS, im Akku Numme der Zeile)
                        ; gibt es noch einen Datensatz ?
```

```
$c113 iny               ; zweites Byte lesen (erstes
                          Register, low)
$c114 lda  ($9e),y      ; Nullbyte ?
$c116 bne $c120         ; nein, dann nicht Tabellenende
$c118 iny               ; nächstes Byte lesen
$c119 lda  ($9e),y      ; auch Null?
$c11b bne $c121         ; nein, dann nicht Tabellenende
                        ; Tabellenende erreicht, wenn drei
                          Nullbytes aufeinanderfolgen
$c11d lda  #$00         ; Kennzeichen: Zero-Flag setzen
$c11f rts               ; und Ende
                        ; nächste Rasterzeile suchen
$c120 iny               ; Registeradresse low überspringen
$c121 iny               ; Registeradresse high überspringen
$c122 iny               ; Inhalt überspringen
$c123 lda  ($9e),y      ; nächste Registeradresse low lesen
$c125 bne $c120         ; nicht Null, dann Datensatz dieser
                          Rasterzeile nicht zu Ende
$c127 iny               ; wenn Null, auch Highbyte testen
$c128 lda  ($9e),y      ; auch Null?
$c12a bne $c121         ; nein, dann Datensatz dieser
                          Rasterzeile nicht zu Ende
                        ; zwei Nullbytes kennzeichnen Ende
                          des Datensatzes zu einer
                          Rasterzeile
$c12c iny               ; nächste Rasterzeilen-Nummer holen
$c12d lda  ($9e),y      ; und prüfen
$c12f beq $c11d         ; Null, dann Tabellenende
$c131 rts               ; sonst fertig

                        ; sucht nächsthöhere Rasterzeile
                          (nach Nummer im Akku)
                        ; (Intervall-Schachtelung)
$c132 sta $c197         ; minimaler Wert
$c135 lda  #$ff         ; maximaler Wert 255
$c137 sta $c198         ; also noch keine Grenze nach oben
                        ; sucht Rasterzeile, die zwischen den
                          beiden Grenzen liegt
$c13a ldy  #$00         ; ab Tabellenanfang suchen
$c13c jsr $c10f         ; Rasterzeile suchen
$c13f beq $c155         ; nicht gefunden, Tabellenende
$c141 cmp $c197         ; größer gesuchtem Minimalwert ?
$c144 bcc $c150         ; zu klein, dann ungültig
$c146 beq $c150         ; gleich, dann ebenfalls nicht gültig
                        ; neue Rasterzeile war größer als
                          Minimalwert
$c148 cmp $c198         ; mit Maximalwert vergleichen
$c14b bcs $c150         ; größer gleich, dann ungültig
                        ; Rasterzeile, deren Nummer (im Akku)
                          zwischen den Grenzwerten liegt,
                          wurde gefunden
$c14d sta $c198         ; Nummer als neuen Maximalwert merken
                        ; weiter suchen
$c150 jsr $c113         ; gibt es noch eine Rasterzeile in
                          der Tabelle?
$c153 bne $c141         ; ja, probieren, ob sie besser paßt
                        ; Suche beendet
$c155 lda $c198         ; wurde eine zufriedenstellende
                          Zeile gefunden?
$c158 cmp  #$ff         ; Maximalwert jemals geändert ?
$c15a bne $c15f         ; ja, dann wurde etwas gefunden
$c15c lda $c197         ; sonst als Alternative bisherigen
                          Minimalwert
$c15f rts               ; verwenden und fertig
                        ; Dieses RTS wird auch am Ende jeder
                          Routine angesprungen, wenn nicht
                          als neue Basicbefehle installiert
                          (siehe $c075)
                        ; Befehl CLEAR (SYS 49504)
$c160 sei               ; alle Interrupts (System/Timer und
                          Raster) sperren
$c161 lda  #$00         ; Rasterinterrupts sperren
                          (Bit 7 Null)
$c163 sta $d01a         ; in IMR schreiben
```

Listing 3. Der komplette dokumentierte Quellcode des »Raster-Utility«

```
$c166 ldx   #$31     ; IRQ-Vektor
$c168 ldy   #$ea     ; auf alten Wert $ea31
$c16a stx $0314      ; stellen
$c16d sty $0315      ; und Routine abschalten
$c170 cli            ; alle Interrupts wieder einschalten
$c171 jmp($c193)     ; fertig (siehe $c075)

                     ; Befehl DELETE (arbeitet fehlerhaft,
                     SYS 49524)
$c174 jsr $b79b      ; Nummer der zu löschenden
                     Rasterzeile holen (nach X)
$c177 stx $c196      ; und merken
$c17a ldy   #$00     ; ab Tabellenanfang suchen
$c17c jsr $c10f      ; gibt es diese Rasterzeile
                     in Tabelle?
$c17f beq $c18b      ; Tabellenende, dann fertig
                     (nicht gefunden)
$c181 cmp $c196      ; gefundene Rasterzeile = gesuchte
                     Zeile ?
$c184 beq $c18c      ; ja, dann löschen
$c186 jsr $c113      ; sonst weitersuchen
$c189 bne $c181      ; gibt es noch eine Rasterzeile?
$c18b rts            ; nein, dann fertig

$c18c lda   #$00     ; Löschkennung: Nullbyte
$c18e sta  ($9e),y   ; Rasterzeileninformation löschen
                     ; da hier das Tabellenende-
                     Kennzeichen gesetzt wird, werden
                     auch alle Rasterzeile-Daten
                     gelöscht, die nach der Definition
                     der jetzt gelöschten Zeile
                     definiert wurden
$c190 jmp($c193)     ; fertig (siehe $c075)
                     ; Ende des Maschinenprogramms

                     ; Varibalen
$c193 .word $c15f    ; Pointer zum Verlassen der Routinen
                     (siehe $c075)
$c195 .byte $ff      ; Zeiger auf Tabellenposition
$c196 .byte $ff      ; Nummer der gesuchten Rasterzeile
$c197 .byte $ff      ; Minimalwert für Suche
$c198 .byte $ff      ; Maximalwert für Suche
$c199 .byte $ff      ; ab hier beginnt der Datenspeicher
                     ; Ende des Files
                     ; Zeropage-Speicherstellen:
$9e/9f: Zeiger in Datenfeld
$f7/f8: Zeiger auf das Register
```

Listing 3. (Schluß)

```
Name : borderless             c738 c786
-----------------------------------------
c738 : 78 a9 60 a0 c7 8d 14 03   f0
c740 : 8c 15 03 a9 00 8d ff 3f   38
c748 : a9 1b 8d 11 d0 a9 33 8d   47
c750 : 12 d0 a9 01 8d 1a d0 a9   95
c758 : 7f 8d 0d dc 58 60 60 60   48
c760 : a9 01 8d 19 d0 ad 12 d0   75
c768 : c9 33 f0 0d a9 17 8d 11   54
c770 : d0 a9 33 8d 12 d0 4c 31   cf
c778 : ea a9 18 8d 11 d0 a9 fa   23
c780 : 8d 12 d0 4c 31 ea 00 00   3e
```

Listing 4. »Borderless« schaltet kurzerhand den Bildschirmrahmen ab und erlaubt Sprites im Rahmen. Bitte mit dem MSE (Seite 159) eingeben.

```
                     ; verlängerter Bildschirm
                     ; für Sprites im Rahmen
                     ; von Nikolaus Heusler

$c738 sei            ; Interrupt verbieten
$c739 lda   #$60     ; IRQ-Vektor auf $c760
$c73b ldy   #$c7     ; Highbyte
$c73d sta  $314      ; verbiegen
$c740 sty  $315      ; (in Vektor 788/789 schreiben)
$c743 lda   #$00     ; Schattenbereich löschen
                     (letztes Byte der VIC-Bank)
$c745 sta $3fff      ; !! liegt mitten im Basicspeicher !!
$c748 lda   #$1b     ; nur Rasterirq bei Zeile 0 - 255
                     zulassen
$c74a sta $d011      ; VIC-Kontrollregister 1
$c74d lda   #$33     ; nächster Raster-IRQ bei Zeile 51
$c74f sta $d012      ; (dieser Wert ist wichtig !)
$c752 lda   #$01     ; Raster-IRQ einschalten
$c754 sta $d01a      ; IMR
$c757 lda   #$7f     ; System-IRQ ausschalten
$c759 sta $dc0d      ; Timer-Register CIA 1
$c75c cli            ; IRQ einschalten
$c75d rts            ; fertig
$c75e rts            ; hier könnte ein JMP .... eingebaut
$c75f rts            ; werden
                     ; neue IRQ-Routine
$c760 lda   #$01     ; Raster-Register
$c762 sta $d019      ; löschen (nächsten IRQ erlauben)
$c765 lda $d012      ; welche Rasterzeile war es ?
$c768 cmp   #$33     ; Nummer 51 (oben) ?
$c76a beq $c779      ; ja, dann entsprechend verzweigen
                     ; IRQ durch Rasterzeile 250
$c76c lda   #$17     ; Bildschirm vergrößern
$c76e sta $d011      ; in Kontrollregister 1
$c771 lda   #$33     ; nächster IRQ bei Rasterzeile 51
$c773 sta $d012      ; (dieser Wert ist wichtig !)
$c776 jmp $ea31      ; und mit System-IRQ weitermachen
                     ; IRQ durch Rasterzeile 51
$c779 lda   #$18     ; Bildschirm verkleinern
$c77b sta $d011      ; Kontrollregister 1
$c77e lda   #$fa     ; nächste Rasterzeile: Nr. 250
                     (wichtig)
$c780 sta $d012      ; löst IRQ aus
$c783 jmp $ea31      ; und weiter wie normal
                     ; Programmende
```

Listing 5. So funktioniert Listing 4 – hier der Quellcode

```
1 IF PEEK(51000)<>120 THEN LOAD"BORDERLESS
  ",8,8                                      <181>
2 POKE 56,63:CLR                             <138>
3 SYS 65409:SYS 51000                        <152>
4 POKE 53280,12:POKE 53281,11                <187>
5 V=5:N=172:J=1:K=2                          <140>
10 POKE 53269,3:POKE 2040,11:POKE 2041,11    <021>
11 POKE 53288,0                              <065>
12 FOR I=.TO 62:POKE 704+I,RND(0)*256:NEXT   <233>
13 PRINT"{CLR}"TAB(6)"{LIG.BLUE}OBERE GREN
   ZE TEXTBILDSCHIRM                         <110>
14 PRINT"{22DOWN}"TAB(6)"UNTERE GRENZE TEX
   TBILDSCHIRM{LIG.RED,HOME}                 <121>
15 PRINT"{8DOWN,2SPACE}DEMO 'BORDERLESS':
   SPRITES IM RAHMEN                         <191>
16 PRINT"{DOWN,3SPACE}NIKOLAUS HEUSLER, 19
   89 -- SH 64'ER                            <191>
17 PRINT"{DOWN,SPACE}WEGEN DER SPRITES FLA
   CKERT ES ETWAS...                         <020>
18 PRINT"{DOWN,6SPACE}SORRY...{2SPACE}IST
   HALT NUR BASIC                            <060>
20 FOR Y=.TO 255 STEP 5                      <004>
22 POKE 53249,Y                              <029>
24 X=SIN(A)*148+172                          <071>
26 POKE 53264,X/256:POKE 53248,X AND 255     <063>
28 A=A+.05                                   <247>
30 POKE 53251,((COS(A*7)*17)+12)AND 255      <155>
32 POKE 53250,N                              <213>
34 N=N+V:IF N<94 OR N>250 THEN V=-V          <198>
36 POKE 16383,2↑K                            <031>
38 K=(K+J)AND 7:W=W+1:IF W=20 THEN W=.:J=-
   J                                         <247>
40 NEXT                                      <050>
50 GOTO 20                                   <236>
```

Listing 6. Demoprogramm zu Listing 4 (dieses wird automatisch nachgeladen)

Das erste Lebenszeichen

Was passiert in den ersten Sekunden nach dem Einschalten Ihres C64? Woher weiß der Prozessor, was er zu tun hat? Was ist ein Reset-Taster, was ein Modul-Start? Auf diese Fragen finden Sie auf dieser Seite ausführlich Antwort.

Haben Sie sich nicht auch schon Gedanken darüber gemacht, warum sich Ihr C64 nach dem Einschalten für kurze Zeit »totstellt«? Brauchen die Chips erst eine kurze Vorwärmzeit? Muß der Prozessor erst einmal wach werden?

Nein, während dieser Phase, in der der Computer zwar eingeschaltet ist, aber noch kein Lebenszeichen (sprich »READY.«-Meldung) von sich gibt, wird ein sogenannter »Reset« ausgeführt. Dieser Reset ist für den C64 nicht eine »Anwärmphase«, sondern eine höchst arbeitsintensive und interessante Routine. Schauen wir uns einmal an, was in unserem C64 nach dem Einschalten so alles vor sich geht.

Wenn Sie Ihren C64 einschalten, so werden die Bausteine im Computer nahezu schlagartig mit Spannung versorgt. Würde die CPU sofort losstürmen, würde sie sofort wieder in den Tiefen des C64 verschwinden. Daran hindert sie jedoch ein Signal am Reset-Eingang. Er liegt auf logisch »0« und lähmt somit den Prozessor. Zu diesem Zeitpunkt wird ein spezieller Timer-Baustein in Gang gesetzt, der die Reset-Leitung kontrolliert. Er wartet, bis sich die Betriebsspannung vollständig aufgebaut und stabilisiert hat. Erst nach Ablauf der hardwaremäßig vorgegebenen Zeitspanne wechselt der Timer seinen Pegel und gibt somit den Prozessor frei, der bis dahin noch nicht eine einzige Operation gemacht hat.

Doch dieser Reset (es gibt noch andere) betrifft nicht nur die CPU. Drei weitere Bausteine bekommen ihn auf die gleiche Weise zu spüren: der Sound-Chip (welch ein Lärm ohne Reset) und die zwei Port-Bausteine (CIAs). Primär betrifft der Einschalt-Reset also vier Bausteine. Doch nur beim Prozessor geht es aktiv weiter, die anderen drei werden nur in eine bestimmte Grundposition »gefahren«, um den Prozessor und uns nicht weiter zu stören.

Am Anfang kommt ein Reset

Wichtig ist, daß der Prozessor bei seinem ersten Zugriff das ROM vorfindet, denn nach Freigabe der Reset-Leitung führt die CPU einen bestimmten Befehl aus. Dafür sorgt der Prozessor selber, indem er an seinen Ausgangsports Basic, Kernel und I/O-Bereich selektiert. Der erste Befehl ist ein indirekter Sprung nach $FFFC, fast an das Ende des Speicherbereichs. Das heißt, der Prozessor »sieht nach«, welche 2-Byte-Adresse sich dort befindet, um sofort dorthin zu springen. Beim C64 steht dort die Adresse $FCE2 (64738). Und dann geht es richtig los.

Zunächst wird der Prozessor gegen normale Interrupts abgesichert (Einen NMI, also RUN/STOP RESTORE, nimmt er nach wie vor an, stürzt aber dabei ab.), der Stack-Pointer wird zurückgesetzt und das Dezimal-Flag (für uns nicht so wichtig, es ist hier nur der Vollständigkeit halber erwähnt) wird gelöscht. Dann prüft der Computer, ob sich ein Modul im Expansions-Port befindet. Diese Routine befindet sich ab $FD02. Er vergleicht die Adressen $8004 bis $8008, ob deren Inhalt den Speicherstellen $FD10 bis $FD14 entspricht. Sie enthalten die Zeichen »CBM80«.

Läuft der Modul-Vergleich erfolgreich ab, wird wieder ein indirekter Sprung ausgeführt, diesmal nach $8000. Es wird also auf die dort vorgefundene Adresse verzweigt. Allerdings erkennt der Computer nicht das Modul, sondern lediglich die Meldung »CBM80«. So ist es möglich, einen solchen Modul-Start zu simulieren, indem man die Modul-Identifikation einfach an die entsprechende Adresse (ab $8004) schreibt und den Vektor bei $8000 auf eine eigene Routine verzweigt. Bei professionellen Spielen wird auf diese Weise der Reset unterdrückt. Dafür bieten einige Speeder wie zum Beispiel SpeedDos, PrologicDos, DolphinDos und das 64'er-Dos die Möglichkeit, während des Resets durch Drücken einer Taste (<SPACE> oder <CTRL>) den Modulstart zu unterbinden. Dies ist allerdings eine Veränderung der Systemsoftware, die im Commodore-Betriebssystem nicht besteht.

Findet der Computer keinen Modul-Start vor, dann wird als nächstes das Register 22 des VIC auf Null gesetzt – das sehen Sie, wenn Sie einen Reset-Taster haben, am Abschneiden des rechten und linken Bildrandes.

Reset sorgt für Ruhe und Ordnung

Im darauffolgenden Schritt, einem Unterprogramm ab $FDA3, werden die Portbausteine und der Sound-Chip, die ja auch einen Reset durchgeführt haben, abgehandelt. Bei letzterem wird nur die Lautstärke auf Null gesetzt, für den Fall, daß der Reset softwaremäßig ausgelöst wurde. In den CIAs dagegen werden die Timer für den Interrupt vorbereitet und gesetzt, die Datenrichtungs-Register, die CIA-Ports und sowie der Prozessor-Port definiert.

Jetzt erst kommt der C64 dazu, seinen Arbeitsspeicher zu testen und zu initialisieren. Die Routine dazu steht ab $FD50 und ist zu etwa 98 Prozent verantwortlich für die Dauer des Resets. Der Bereich von $0002 bis $03FF wird komplett gelöscht, der gesamte Speicher von $0400 bis $9FFF wird durch zweimaliges Schreiben und Auslesen auf Funktion getestet. Daraufhin werden die Ober- und Untergrenze des Basic-Bereiches definiert und das Video-RAM (der Bildschirm) nach $0400 gelegt.

Hardware und Ein-/Ausgabe-Vektoren sind die nächste Station des Resets. Damit sind die Zeiger ab $0314 gemeint. Deren Inhalte werden von einer Routine ab $FD15 einfach aus einer Tabelle ab $FD30 kopiert.

Die letzte Station des Resets ist die Initialisierung des Video-Controllers. Eine Tabelle ab $ECB9 wird nach $D000 kopiert. Dadurch wird der VIC initialisiert. Danach wird die Tastatur-Eingabe vorbereitet, der Bildschirm gelöscht und nochmals der Interrupt-Timer gesetzt. Nachdem dann der Interrupt schließlich auch noch zugelassen wurde, wird der Basic-Kaltstart, durch einen indirekten Sprung nach $A000, ausgeführt.

Wir haben vorher angesprochen, daß es mehrere Möglichkeiten gibt, einen Reset auszulösen. Den Einschalt-Reset können Sie auslösen, indem Sie die Reset-Leitung kurzzeitig auf Masse ziehen, beispielsweise durch einen Taster. Eine Bauanleitung dafür finden Sie im 64'er, Ausgabe 6/85.

Einige Erweiterungen beinhalten Reset-Befehle, wie COLD bei Simons Basic, die dann einen Reset softwaremäßig auslösen. Dem entspricht ein SYS 64738 im normalen Basic. Allerdings ist dies kein »sauberer« Reset, da nicht automatisch auf das ROM geschaltet wird und auch der SID und die CIAs nicht davon betroffen sind. Der Einsprung ist allerdings derselbe, also mit Modul-Test und Initialisierung aller wichtigen Register, so daß Sie hier beliebig experimentieren können. Natürlich lassen sich die diversen Einsprünge auch unabhängig voneinander verwenden. So löst zum Beispiel SYS 64763 nur einen Video-Reset mit Basic-Kaltstart aus, ohne eventuell verstellte Interrupt-Vektoren zu korrigieren.

(Markus Ohnesorg/ef)

Tips und Tricks für Basic-Programmierer

Mit den folgenden kleinen Programmen geben wir Ihnen einige Hilfsmittel an die Hand, die Sie in Ihre Programme einbinden können, um sie schneller und effektiver zu machen.

Vielleicht geht es Ihnen wie mir. Sie kommen morgens nicht aus Ihrem Bett, verfluchen jeden Tag aufs neue Ihren Wecker, oder verbrauchen die Nerven Ihres Partners, der verzweifelt versucht, Sie zu wecken. Falls dies der Fall sein sollte, haben Sie sicher schon einmal versucht, sich durch Ihren Computer wecken zu lassen. Sollten Sie zu diesem Zweck die Variable TI$ (die sich ja geradezu aufdrängt) verwendet haben, so würden Sie zwar an diesem Morgen geweckt, jedoch sicher nicht zu dem Zeitpunkt, den Sie sich vorgestellt hatten. Die Variable TI$ hat leider die unangenehme Eigenschaft, eine recht ungenaue Uhr zu sein. Der maximale Fehler liegt bei 30 Minuten pro Tag. Aber zum Glück hat unser Computer, wie so oft, auch hierfür eine Lösung parat. Es handelt sich hierbei um die beiden CIAs (Complex Interface Adapter), welche je eine Echtzeituhr enthalten. Diese Echtzeituhr erhält ihren Takt aus der Netzfrequenz, die Variable TI$ hingegen wird von der Interruptroutine versorgt. Dieser Tatsache verdanken wir es, daß die CIA-Uhr eine so große Langzeitgenauigkeit vorweisen kann. Die Echtzeituhr hält aber noch eine weitere Überraschung für uns bereit. Es besteht die Möglichkeit, eine Alarmzeit anzugeben. Sobald diese Zeit erreicht wird, löst das CIA einen IRQ (Interrupt, beziehungsweise CIA 2 einen NMI = unmaskierter Interrupt) aus, aber dazu später noch mehr. Wie die Register der CIA-Bausteine belegt sind, können Sie aus Tabelle 1 entnehmen. Die Echtzeituhr belegt die Register 8 bis 11.

Die Zeit starten

Die CIA-Uhren sind im BCD-Format organisiert. Dadurch sparen wir uns zwar in Maschinensprache das lästige Umrechnen, von Basic aus ist die Umwandlung leider nicht zu umgehen. Im BCD-Format werden Zahlen zwar immer noch Bitweise gespeichert, jedoch jede Ziffer einzeln. Mit 8 Bit lassen sich also zwei Ziffern (2 mal 4 Bit, 00 bis 99) darstellen. Um nun eine Zahl, bestehend aus zwei Ziffern, in das BCD-Format umzuwandeln, wird die erste Ziffer (Wertigkeit 10) mit 16 multipliziert und zu der zweiten Ziffer (Wertigkeit 1) addiert.

Aber nun wieder zu unseren CIAs. Die Basisadresse des CIA 1 ist 56320 (CIA 2 = 56576). Eine kurze Registerbeschreibung entnehmen Sie bitte Bild 1. Zum Stellen der Uhr werden einfach die entsprechenden BCD-Werte in die jeweiligen Register geschrieben. Sobald Sie das $^1/_{10}$ Sekunden-Register überschreiben, startet die Uhr (überschreiben Sie jedoch die Stundenregister, stoppt sie). An Bit 7 in Register 11 können Sie außerdem erkennen, ob es sich um eine AM- oder PM-Zeit handelt (AM = Vormittag, PM = Nachmittag). Bitte vergessen Sie nicht, Bit 7 in Register 14 zu setzen (Netzfrequenz 50 HZ (60 HZ)). Für den Fall, daß Sie Ihre Uhr auch lesen möchten (soll ja vorkommen), bietet das CIA eine weitere Besonderheit. Sobald Sie eines der Register lesen, wird die komplette Uhrzeit in einen Zwischenspeicher übernommen. Durch diesen Kniff können Sie in aller Ruhe die Daten auslesen, ohne daß eine Veränderung des Zwischenspeichers eintritt. Sobald Register 8 angesprochen wird, gibt das CIA den Speicher wieder frei. In Listing 1 wartet ein Basic-Programm auf Sie, mit welchem Sie die Zeit stellen und ablesen können (Start mit RUN, um sie zu stellen und RUN 200, um sie zu stoppen). Listing 2 enthält eine kurze Assemblerroutine, die in die Interruptroutine eingefügt wird und Ihnen die Uhrzeit laufend auf dem Bildschirm ausgibt. Die eigentliche Assemblerroutine liegt bei Adresse 830. Um das Programm zu aktivieren, befindet sich bei Adresse 900 ein Programmteil, welcher den Interruptvektor auf das Hauptprogramm legt (SYS 900 startet die Anzeige der Uhr).

Der Wecker im Computer

Neben einer Uhr hat das CIA natürlich noch weitere Aufgaben. Es löst zum Beispiel das Interruptsignal aus, durch welches die bereits erwähnte Interruptroutine angesprungen wird. Dieses Ereignis tritt bei Unterlauf des Timers (die CIAs haben neben der Uhr noch zwei Timer) automatisch alle $^1/_{60}$ Sekunden auf. Wir haben allerdings die Möglichkeit zu einem von uns vordefinierten Zeitpunkt einen zusätzlichen Interrupt auszulösen. Zu diesem Zweck müssen wir eine Zeit angeben, die wir als Alarmzeit heranziehen. Die Alarmzeit wird wie die normale Uhrzeit in die Register 8 bis 11 geschrieben, jedoch muß hierfür das Bit 7 in Register 15 gesetzt sein (Rücksetzen nicht vergessen). Stimmen nun die Uhrzeit und die von uns eingestellte Alarmzeit überein, wird ein zusätzlicher IRQ ausgelöst und der Programmzeiger springt zu der

CIA (6526)

		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
Register 0 $DC00	Port A	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	56320
Register 1 $DC01	Port B	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	56321
Register 2 $DC02	Datenrichtungsregister A	DPA7	DPA6	DPA5	DPA4	DPA3	DPA2	DPA1	DPA0	56322
Register 3 $DC03	Datenrichtungsregister B	DPB7	DPB6	DPB5	DPB4	DPB3	DPB2	DPB1	DPB0	56323
Register 4 $DC04	Timer A Low-Byte	TAL7	TAL6	TAL5	TAL4	TAL3	TAL2	TAL1	TAL0	56324
Register 5 $DC05	Timer A High-Byte	TAM7	TAM6	TAM5	TAM4	TAM3	TAM2	TAM1	TAM0	56325
Register 6 $DC06	Timer B Low-Byte	TBL7	TBL6	TBL5	TBL4	TBL3	TBL2	TBL1	TBL0	56326
Register 7 $DC07	Timer B High-Byte	TBM7	TBM6	TBM5	TBM4	TBM3	TBM2	TBM1	TBM0	56327
Register 8 $DC08	Uhr 1/10 Sekunde	0	0	0	0	T0	T4	T2	T1	56328
Register 9 $DC09	Uhr Sekunde	0	SM4	SM2	SM1	SL0	SL4	SL2	SL1	56329
Register 10 $DC0A	Uhr Minuten	0	MH4	MH2	MH1	ML0	ML4	ML2	ML1	56330
Register 11 $DC0B	Uhr Stunden	PM	0	0	MH	ML0	ML4	ML2	ML1	56331
Register 12 $DC0C	Serial Data Register	S7	S6	S5	S4	S3	S2	S1	S0	56332
Register 13 $DC0D	Interrupt Control Register	IRQ	0	0	FLG	SP	PLRM	T0	TA	56333
Register 14 $DC0E	Control Register A	S0 HZ	SP MODE	IN MODE	LOAD	RUNMODE	OUTMODE	PB ON	START	56334
Register 15 $DC0F	Control Register B	ALARM	IN MODE	IN MODE	LOAD	RUNMODE	OUTMODE	PB ON	START	56335

Tabelle 1. Registerbelegung der CIA

in den IRQ-Vektoren angegebenen Adresse. Die Unterscheidung zwischen einem Interrupt durch den Unterlauf eines Timers oder durch Erreichen der Alarmzeit wird durch Register 13 möglich. In diesem Register wird bei Auftreten des IRQ das zuständige Bit (siehe Tabelle 1) gesetzt. Zum besseren Verständnis erwartet Sie in Listing 3 eine kleine Routine, welche bei Erreichen der Alarmzeit die Farbe des Bildschirmrahmens verändert. Diese Routine wird ebenfalls mit SYS 900 gestartet.

Bildschirm mal zwei

Wäre es nicht praktisch, auf dem Monitor zwei voneinander unabhängige Bildschirmseiten unterzubringen? Würde sich ein HiRes-Bild und der dazugehörige Text nicht ungeheuer gut machen? Wenn Ihnen diese oder ähnliche Gedanken schon einmal durch den Kopf gingen, dann sind Sie hier genau richtig. Mittels des VICs (Videocontroller) ist es unserem Computer nicht nur möglich, Zahlen oder Buchstaben auf den Bildschirm zu bringen, er sorgt auch für die HiRes-Grafik, die Sprites und die Verwaltung von 16 KByte Speicherplatz. Da er nebenbei auch noch für die Erzeugung eines normgerechten PAL-Signals zuständig ist, muß er in jedem Augenblick genau wissen, an welcher Stelle sich der Elektronenstrahl Ihres Monitors gerade befindet. Mit einer kleinen Assemblerroutine und ein wenig Bastelei an den Interruptvektoren und -registern unseres Computers können Sie in dem Augenblick, in dem sich der Elektronenstrahl an einer von Ihnen bestimmten Position befindet, zum Beispiel von einem HiRes- auf einen Textbildschirm umschalten (und umgekehrt). Da wir dadurch dem Computer keine Chance lassen, eine Bildschirmseite zu vollenden und einen Bildwechsel vorzunehmen, erscheinen zwei (oder mehrere) Bilder gleichzeitig. Die Trennung dieser Bilder ist abhängig von der Position, an welcher die Umschaltung erfolgt. Die Assemblerroutine in Listing 4 ermöglicht die Darstellung von zwei Textbildschirmen (1024–2023 und 2024–3043) gleichzeitig. In der Speicherstelle 648 können Sie festlegen, auf welchem Bildschirmteil Ihre Eingaben erscheinen sollen. Alle Ausgaben auf den Bildschirm werden ebenfalls in die Bildschirmseite geschrieben, deren Adresse Sie in Speicherstelle 648 angegeben haben. Der Wert, den Sie an diese Adresse POKEn müssen, errechnet sich wie folgt: Adresse des Bildschirmspeichers durch 256. Sobald Sie sich für eine Bildschirmhälfte entschieden haben, können Sie das Programm mit SYS 900 starten (Hier wird der Interruptvektor gesetzt, das eigentliche Programm befindet sich ab Speicherplatz 830. Oberer Bildschirm POKE 648,4, unterer Bildschirm POKE 648,8).

Der VIC hält für uns die Möglichkeit bereit, bei Auftreten eines bestimmten Ereignisses einen Interrupt (IRQ) auszulösen. Zu diesem Zweck stehen die Register 25 und 26 zur Verfügung. Die genaue Belegung dieser Register entnehmen Sie bitte Tabelle 2. Falls mindestens ein Bit aus Register 25 und Register 26 übereinstimmt, wird ein IRQ ausgelöst und der interne Programmzähler springt zu der im IRQ-Vektor (788/789) angegebenen Adresse. Sobald wir diesen Zeiger auf ein eigenes Programm zeigen lassen, wird bei jedem IRQ zuerst unser Programm angesprungen. Das Ende unseres Programmes sollte immer einen unbedingten Sprung in die eigentliche Interruptroutine beinhalten. In unserem Fall müssen wir die Interruptmaske (Register 26) so verändern, daß Register 18 als Auslöser anerkannt wird. Um eine Auslösung an einer bestimmten Stelle des Rasterstrahls zu erreichen, wird in Register 18 (17) die entsprechende Position angegeben. Register 18 hat folglich zwei Aufgaben: 1. wird es gelesen, so gibt es die aktuelle Position des Rasterstrahls an. 2. wird es beschrieben, so gilt der Wert, welcher in das Register geschrieben wurde als die Position des Strahls, an der der IRQ ausgelöst (und Bit 1/Register 25 ist gesetzt) wird. Als letztes Problem bleibt uns noch das CIA. Dieser Chip ist im Normalfall derjenige, welcher die IRQs auslöst. Für unseren Zweck benötigen wir jedoch nur das Interruptsignal, das vom VIC ausgelöst wurde. Aus diesem Grund wird zu Beginn der

```
10 C=56328                                     <226>
20 POKE C+14,PEEK(C+14) OR 128                 <163>
30 POKE C+15,PEEK(C+15) AND 127                <136>
40 INPUT "ZEIT HHMMSS";A$                      <220>
50 IF LEN(A$)<>6 THEN 40                       <057>
60 H=VAL(LEFT$(A$,2))                          <026>
70 M=VAL(MID$(A$,3,2))                         <239>
80 S=VAL(RIGHT$(A$,2))                         <154>
100 IF H>12 THEN H=H+68                        <001>
110 POKE C+3,16*INT(H/10)+H-INT(H/10)*10       <211>
130 POKE C+2,16*INT(M/10)+M-INT(M/10)*10       <108>
150 POKE C+1,16*INT(S/10)+S-INT(S/10)*10       <090>
160 POKE C,0                                   <224>
180 PRINT"{CLR}"                               <168>
200 C=56328                                    <160>
210 H=PEEK(C+3):M=PEEK(C+2):S=PEEK(C+1):T=
    PEEK(C)                                    <230>
230 F=0:IF H>32 THEN H=H-128:F=1               <013>
240 H=INT(H/16)*10+H-INT(H/16)*16:IF F=0 T
    HEN 280                                    <251>
250 IF H=12 THEN 290                           <126>
260 H=H+12                                     <250>
280 IF H=12 THEN H=0                           <167>
290 M=INT(M/16)*10+M-INT(M/16)*16              <173>
300 S=INT(S/16)*10+S-INT(S/16)*16              <131>
320 T$=RIGHT$("0"+MID$(STR$(H),2,2),2)+":"     <254>
340 T$=T$+RIGHT$("0"+MID$(STR$(M),2,2),2)+
    ":"                                        <023>
360 T$=T$+RIGHT$("0"+MID$(STR$(S),2,2),2)+
    ":"                                        <067>
380 T$=T$+RIGHT$(STR$(T),1)                    <166>
400 PRINT"{HOME,3DOWN}";T$                     <050>
420 GOTO 210                                   <118>
```

Listing 1. Basic-Programm zum Stellen der Uhr

```
NAME : LISTING 2 MC     033E 03A3
----------------------------------------
033E : A2 03 A0 1E BD 08 DC 85   E8
0346 : FB 29 F0 18 6A 6A 6A 6A   8D
034E : 69 30 EA 99 00 04 A5 FB   6C
0356 : 29 0F 69 30 C8 99 00 04   C9
035E : C8 A9 3A 99 00 04 C8 CA   95
0366 : E0 00 D0 D8 AD 08 DC 69   F7
036E : 2F 99 00 04 4C 31 EA EA   BA
0376 : EA EA EA EA EA EA EA EA   75
037E : EA EA EA EA EA EA 78 A9   30
0386 : 3E 8D 14 03 A9 03 8D 15   03
038E : 03 58 60 EA EA EA EA EA   BA
0396 : EA EA EA E6 FC D0 F4 E6   85
039E : FD A5 FD C9 08 00 00 00   A7
```

Listing 2. Interruptroutine Uhrzeit

```
NAME : LISTING 3 MC     033E 03A3
----------------------------------------
033E : A2 03 A0 1E BD 08 DC 85   E8
0346 : FB 29 F0 18 6A 6A 6A 6A   8D
034E : 69 30 EA 99 00 04 A5 FB   6C
0356 : 29 0F 69 30 C8 99 00 04   C9
035E : C8 A9 3A 99 00 04 C8 CA   95
0366 : E0 00 D0 D8 AD 08 DC 69   F7
036E : 2F 99 00 04 AD 0D DC C9   35
0376 : 81 F0 03 EE 20 D0 4C 31   2A
037E : EA EA EA EA EA EA 78 A9   30
0386 : 3E 8D 14 03 A9 03 8D 15   03
038E : 03 58 60 EA EA EA EA EA   BA
0396 : EA EA EA E6 FC D0 F4 E6   85
039E : FD A5 FD C9 08 00 00 00   A7
```

Listing 3. Interruptroutine Alarmzeit

```
NAME : LISTING 4 MC     033E 03AD
----------------------------------------
033E : AD 19 D0 A2 FF 8E 19 D0   7B
0346 : 29 80 C9 80 F0 06 4C 31   05
034E : EA EA EA EA AD 18 D0 C9   38
0356 : 15 F0 07 A9 15 A2 96 18   CB
035E : 90 04 A9 25 A2 00 8D 18   90
0366 : D0 8E 12 D0 4C 81 EA EA   6E
036E : EA EA EA EA EA EA EA EA   6D
0376 : EA EA EA EA EA EA EA EA   75
037E : EA EA EA EA EA EA 78 A9   30
0386 : 3E A2 03 8D 14 03 8E 15   46
038E : 03 A9 F9 8D 1A D0 AD 11   97
0396 : D0 29 7F 8D 11 D0 A9 96   F8
039E : 8D 12 D0 58 60 EA EA EA   52
03A6 : EA EA EA EA EA EA EA 00   CF
```

Listing 4. Bildschirmteilung

Register	Adresse dezimal	Adresse hex	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	53248	$D000	X-Position Sprite Nr. 0							
1	53249	$D001	Y-Position Sprite Nr. 0							
2	53250	$D002	X-Position Sprite Nr. 1							
3	53251	$D003	Y-Position Sprite Nr. 1							
4	53252	$D004	X-Position Sprite Nr. 2							
5	53253	$D005	Y-Position Sprite Nr. 2							
6	53254	$D006	X-Position Sprite Nr. 3							
7	53255	$D007	Y-Position Sprite Nr. 3							
8	53256	$D008	X-Position Sprite Nr. 4							
9	53257	$D009	Y-Position Sprite Nr. 4							
10	53258	$D00A	X-Position Sprite Nr. 5							
11	53259	$D00B	Y-Position Sprite Nr. 5							
12	53260	$D00C	X-Position Sprite Nr. 6							
13	53261	$D00D	Y-Position Sprite Nr. 6							
14	53262	$D00E	X-Position Sprite Nr. 7							
15	53263	$D00F	Y-Position Sprite Nr. 7							
16	53264	$D010	Sprite 7, msb X-Position	Sprite 6, msb X-Position	Sprite 5, msb X-Position	Sprite 4, msb X-Position	Sprite 3, msb X-Position	Sprite 2, msb X-Position	Sprite 1, msb X-Position	Sprite 0, msb X-Position
17	53265	$D011	msb des Rasterregisters (Reg. 18)	Schaltbit für veränderten Hintergrund-farbmodus 1 = eingeschaltet	Schaltbit für Hochauflösungs-modus 1 = eingeschaltet	Schaltbit für Schirm löschen 0 = gelöscht	Schaltbit für Zeilenzahl 0 = 24 Zeilen 1 = 25 Zeilen	Wert der Zeilenverschiebung in Y-Richtung beim Smooth Scrolling		
18	53266	$D012	Rasterregister. Dazu kommt das msb in Bit 7, Register 17							
19	53267	$D013	Lichtgriffel X-Position							
20	53268	$D014	Lichtgriffel Y-Position							
21	53269	$D015	Ein- und Ausschalten von Sprites. 0 = Sprite aus, 1 = Sprite an							
			Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
22	53270	$D016	(unbenutzt)		Reset-Bit, muß 0 sein, damit VIC-II-Chip arbeitet	Schaltbild für Mehrfarbmodus 1 = eingeschaltet	Schaltbit für Spaltenzahl 0 = 38 Spalten 1 = 40 Spalten	Wert der Spaltenverschiebung in X-Richtung beim Smooth Scrolling		
			Sprite-Vergrößerung in Y-Richtung. 0 = normale Größe, 1 = doppelte Größe							
23	53271	$D017	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
24	53272	$D018	Startadresse Textbildschirm				Startadresse Zeichengenerator oder HiRes-Bitmap			(unbenutzt)
25	53273	$D019	Interrupt-Flag-Register				Lichtgriffel-Interrupt-Flag	Sprite/Sprite-Kollision	Sprite/Hinter-grund-Kollision	Raster-Interrupt-Flag
26	53274	$D01A	Interrupt-Masken-Register				Lichtgriffel-Interrupt	Sprite/Sprite-Kollision	Sprite/Hinter-grund-Kollision	Raster-Interrupt-Maske
			Sprite/Hintergrund-Prioritätenregister. 0 = Sprite hat Priorität, 1 = Hintergrund hat Priorität							
27	53275	$D01B	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
			Sprite-Mehrfarbmodus-Register. 0 = Normaldarstellung, 1 = Mehrfarbmodus-Darstellung							
28	53276	$D01C	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
			Sprite-Vergrößerung in X-Richtung. 0 = normale Größe, 1 = doppelte Größe							
29	53277	$D01D	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
			Sprite/Sprite-Kollision. 0 = keine Berührung. 1 = Berührung							
30	53278	$D01E	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
			Sprite/Hintergrund-Kollision. 0 = keine Berührung, 1 = Berührung							
31	53279	$D01F	Sprite 7	Sprite 6	Sprite 5	Sprite 4	Sprite 3	Sprite 2	Sprite 1	Sprite 0
32	53280	$D020	(unbenutzt)				Farbe des Bildschirmrahmens			
33	53281	$D021	(unbenutzt)				Hintergrundfarbe Nr. 0 (normale Hintergrundfarbe)			
34	53282	$D022	(unbenutzt)				Hintergrundfarbe Nr. 1			
35	53283	$D023	(unbenutzt)				Hintergrundfarbe Nr. 2			
36	53284	$D024	(unbenutzt)				Hintergrundfarbe Nr. 3			
37	53285	$D025	(unbenutzt)				Sprite-Mehrfarben-Register Nr. 0			
38	53286	$D026	(unbenutzt)				Sprite-Mehrfarben-Register Nr. 1			
39	53287	$D027	(unbenutzt)				Sprite 0, Farbe			
40	53288	$D028	(unbenutzt)				Sprite 1, Farbe			
41	53289	$D029	(unbenutzt)				Sprite 2, Farbe			
42	53290	$D02A	(unbenutzt)				Sprite 3, Farbe			
43	53291	$D02B	(unbenutzt)				Sprite 4, Farbe			
44	53292	$D02C	(unbenutzt)				Sprite 5, Farbe			
45	53293	$D02D	(unbenutzt)				Sprite 6, Farbe			
46	53294	$D02E	(unbenutzt)				Sprite 7, Farbe			

Tabelle 2. Alle Register des Video-Chips auf einen Blick



Routine mittels des IRQ-Flags in Register 25 abgefragt, ob der VIC den IRQ angefordert hat.

Basic-Zeilen in ein laufendes Programm einfügen? Wollen Sie Funktionen austesten oder ein Basic-Programm entwickeln, welches sich selbst verändert? Mit unserem Basic »Listing 5« und der Assemblerroutine aus »Listing 5 MC« können Sie ohne Schwierigkeiten alle beliebigen Daten in Ihr Basicprogramm aufnehmen, ohne das Programm stoppen zu

Nachträglich Befehle einfügen

müssen. Für diese Aufgabe stellt das Betriebssystem unseres Computers einige sehr effektive Routinen zur Verfügung. Unsere Aufgabe besteht nur noch darin, die einzufügenden Daten in den Basic-Eingabepuffer (512–600) zu schreiben und die Routine aufzurufen. Nach Aufruf des Programms wird der Vektor zur Eingabe einer Zeile auf eine Adresse innerhalb unseres Programms gesetzt. Daraufhin wird die komplette Betriebssystemroutine zur Einfügung von Programmzeilen abgearbeitet. Zum Ende dieser Routine würde der Programmzeiger in die Eingabe-Warteschleife springen. Durch die anfängliche Änderung des entsprechenden Vektors wird der Programmzeiger jedoch wieder auf unsere Routine umgelenkt. Hier wird nun der Vektor wieder auf den ursprünglichen Wert verändert und das Basic-Programm erneut gestartet. Wie bereits zu Anfang erwähnt, eignet sich dieser kleine Trick hervorragend zur Eingabe von Funktionen.

Basic optimieren

Sie haben sich sicher schon oft über die schwache Leistung unseres Basic-Interpreters geärgert. Je komplexer die Anforderungen an ein Basic-Programm sind, desto langweiliger wird die Geschwindigkeit der betreffenden Programme. Viele scheuen jedoch den Schritt in Richtung Maschinensprache und ärgern sich weiterhin mit ihren »Schlafmützen«-Programmen. Betrachtet man aber die Arbeitsweise des Interpreters ein wenig genauer, so bieten sich einige Möglichkeiten zur Beschleunigung von Basic-Programmen. Einmal abgesehen von den verwendeten Algorithmen und der Struktur des Programmes gibt es mehrere Tricks, um den Interpreter ein wenig anzutreiben.

Variable

Sobald in einem Basic-Programm eine Variable benötigt wird, beginnt der Interpreter, vom Variablenstart an aufwärts zählend, nach der Variable zu suchen. Je später eine Variable definiert wurde, desto länger dauert der Suchvorgang. Ähnlich verhält es sich bei Konstanten. Werden Konstante im Programm angegeben, so müssen diese bei jedem Durchlauf erneut umgewandelt werden. Definiert man Konstante als Variable, dann fällt die Umrechnung nur einmal an.

(1) Konstante, Variable, und Felder in der Reihenfolge ihrer Zugriffshäufigkeit vordefinieren. Möglichst ein- oder zweistellige Variablennamen verwenden.

Unterprogramme

Unterprogramme müssen vom Interpreter, wie Variablen, erst einmal gesucht werden. Da die Suche mit der niedrigsten Zeilennummer beginnt, findet er Zeilen am Anfang natürlich schneller.

(2) Unterprogramme sollten am Anfang eines Programmes stehen.

Verzweigungen

IF-THEN-Abfragen mit mehreren, durch AND-Verknüpfungen verbundenen, Bedingungen sollten durch Hintereinanderlegen von mehreren IF-THEN-Abfragen verschachtelt werden. Beispiel:

a. IF (A kleiner 100 AND B größer 0) THEN
b. IF A kleiner 100 THEN IF B größer 0 THEN

Version b bricht sofort nach Nichterfüllung der ersten Bedingung ab und ist somit schneller.

(3) IF-THEN-Abfragen mit mehreren Bedingungen verschachteln.

Leerzeichen

REMs und Leerstellen verzögern den Programmlauf, da der Interpreter sie ja ignorieren muß. Je mehr Befehle sich in einer Programmzeile befinden, desto seltener muß der Computer nach neuen Zeilenanfängen suchen.

(4) Programme kompakter verfassen.

Strings

Die Garbage-Collection-Routine des Betriebssystems beseitigt den »String-Müll«. Werden mehrere Strings definiert, fällt auch mehr »Müll« an. Das Tragische an dieser Routine ist die Geschwindigkeit: verdoppelt sich die Anzahl der Strings, vervierfacht (!) sich die Laufzeit. Aus diesem Grund bringt das Anlegen von String-Konstanten auch keinen Zeitgewinn.

(5) So wenig Strings wie möglich benutzen.

Potenzieren

Falls in Ihrem Programm eine ganzzahlige Potenzierung (x^2, x^4..) vorkommt, ist es ratsam, die Potenzierung durch eine Mehrfachmultiplikation zu ersetzen. Da die Potenzierungsroutine des Interpreters auch Potenzierungen mit Brüchen berrechnen kann, ist sie $1^1/_2$ mal langsamer als die entsprechende Multiplikation.

(6) Potenzierung durch Multiplikation ersetzen.

Interrupt

Das CIA 1 löst alle $^1/_{60}$ Sekunden einen IRQ aus, um daraufhin den Programmzeiger in die Interruptroutine springen zu lassen. Hier wird die Tastatur abgefragt, die Zeit erhöht und noch einige andere Dinge erledigt. Die Bearbeitung und Beachtung des Interrupts zweigt natürlich einen Teil der ach so kostbaren Rechenzeit ab. Wenn wir den Interrupt verhindern, solange wir die Tastatur nicht benötigen, so kann auch mit dieser Technik ein Basic-Programm beschleunigt werden. POKE 56333,31 verhindert den IRQ, POKE 56333,159 gibt den IRQ wieder frei.

(7) Sperren des Interrupts.

Probieren Sie es aus. Verändern Sie Ihre alten Programme in der beschriebenen Weise und vergleichen Sie die Geschwindigkeit. Sie werden überrascht sein.

Zeitsparen einmal anders

Basic-Programme werden meist erst durch den Einsatz von Assemblerroutinen zu Programmen mit vernünftiger Laufgeschwindigkeit. Der Sprung in diese Routinen wird häufig durch einen SYS-Befehl realisiert. Die Parameterübergabe wird in diesen Fällen meist durch vorangehende POKE-Befehle bewerkstelligt. Sie sollen sich nun nicht mehr länger mit dieser umständlichen, uneleganten und vor allem langsamen Methode begnügen müssen. In unserem Computer verbirgt sich eine Funktion, die von vielen Handbüchern schlicht vergessen oder nur im Vorübergehen behandelt wird. Es handelt sich hierbei um die »USR-Funktion«. Im Gegensatz zum SYS-Befehl beinhaltet die USR-Funktion bereits eine vollständige Parameterübergabe. Sie kann also genauso benutzt werden wie alle anderen Funktionen des Basic-Interpreters (CHR$(x), ASC(x) ...). Der Vorteil dieser Funktion ist die freie Definition der eigentlichen Operation durch den Programmierer. Aber zunächst einmal die Form der Funktion: x1=USR(x2), wobei x2 der Wert ist, mit welchem die Routine arbeitet und x1 das Ergebnis der Operation darstellt. Die Parameter x1 und x2 können jede beliebige Form annehmen, wie zum Beispiel: Zahl, Variable, Zeichen, String (natürlich muß der Parameter zur jeweiligen Operation passen). x2 wird beim Aufruf der Routine automatisch in den FAC (Fließkommaakkumulator) übernommen. Analog dazu wird der Wert des FAC am Ende des Programmlaufs in x1 geschoben. Die weitere Funktionsweise der USR-Funktion ist ähnlich dem

```
0 REM **** STARTZEILE ****                        <099>
1000 INPUT A$:IF A$=""THEN END                     <197>
1010 FOR AA=0 TO LEN(A$)-1:POKE 513+AA,ASC
     (MID$(A$,AA+1,1)):NEXT:POKE 513+AA,0          <071>
1020 POKE 11,AA:SYS 830                            <100>
```

Listing 5a. Basic-Teil einfügen

```
NAME : LISTING 5 MC      033E 037B
-----------------------------------
033E : A9 55 8D 02 03 A9 03 8D  DA
0346 : 03 03 A9 00 85 7A A9 02  0C
034E : 85 7B A9 30 4C 9C A4 A9  91
0356 : 83 8D 02 03 A9 A4 8D 03  7D
035E : 03 4C 71 A8 EA EA EA EA  80
0366 : EA EA EA EA EA EA EA EA  65
036E : EA EA EA EA EA EA EA EA  6D
0376 : EA EA EA EA EA EA EA EA  75
```

Listing 5b. Maschinen-Teil einfügen

```
NAME : LISTING 6 MC      033E 0353
-----------------------------------
033E : 20 6B E2 20 E2 BA 60 EA  2C
0346 : EA EA EA EA EA EA EA EA  45
034E : EA EA EA EA EA 9C A4 A9  3F
```

Listing 6. USR-Routine

```
NAME : LISTING 7 MC      033C 03F3
-----------------------------------
033C : 20 8D AD A6 2F A5 30 86  51
0344 : 5F 85 60 C5 32 D0 04 E4  BA
034C : 31 F0 1D A0 00 B1 5F C8  ED
0354 : C5 45 D0 06 A5 46 D1 5F  43
035C : F0 17 C8 B1 5F 18 65 5F  4B
0364 : AA C8 B1 5F 65 60 90 D7  16
036C : A2 E2 B6 22 A9 03 4C 45  D4
0374 : A4 C8 B1 5F 18 65 5F 85  0A
037C : 24 C8 B1 5F 65 60 85 25  16
0384 : C8 B1 5F 20 96 B1 85 5F  CC
038C : 84 60 24 0E 30 1F 20 A2  CD
0394 : BB 18 90 04 20 67 B8 18  50
039C : A5 5F 69 05 85 5F 90 02  85
03A4 : E6 60 A4 60 C5 24 90 EC  89
03AC : C4 25 90 E8 60 20 D5 03  A8
03B4 : 20 0C BC 18 A5 5F 69 02  0B
03BC : 85 5F 90 02 E6 60 C5 24  26
03C4 : 90 06 A5 60 C5 25 B0 E4  DF
03CC : 20 D5 03 20 6F B8 4C B4  F3
03D4 : 03 A0 00 B1 5F AA C8 B1  2F
03DC : 5F A8 BA 4C 91 B3 41 52  1C
03E4 : 52 41 59 20 4E 4F 54 20  22
03EC : 46 4F 55 4E C4 00 00 00  45
```

Listing 7. Array-Berechnungen

```
NAME : LISTING 8 MC      033E 035D
-----------------------------------
033E : 20 FD AE 20 9E B7 BA 48  6F
0346 : 20 FD AE 20 9E B7 68 A8  AF
034E : 18 20 F0 FF 20 FD AE 4C  FB
0356 : A4 AA EA EA EA EA EA 17  47
```

Listing 8. Print AT-Simulation

SYS-Befehl beziehungsweise dem Aufruf einer normalen Basic-Funktion. Nach Aufruf der Funktion wird ein Assemblerprogramm angesprungen. Die Adresse dieses Programmes muß jedoch nicht bei jedem Aufruf erneut angegeben werden, sondern wird im sogenannten USR-Vektor (785/786) vordefiniert. Falls Sie die USR-Funktion bereits einmal ausprobieren wollten und als Antwort ein »SYNTAX ERROR« erhielten, so lassen Sie sich bitte davon nicht irritieren. Der USR-Vektor wird, wie alle anderen Vektoren, während des »Start-Resets« auf eine festgelegte Routine des Betriebssystems gelegt. Der Startwert des USR-Vektors zeigt dann auf eine Routine, die einen »SYNTAX ERROR« ausgibt. Aber nun zu unserer ersten Anwendung. Sie finden in Listing 6 eine Routine, die von dem angegebenen Wert den Sinuswert errechnet und diesen mit 10 multipliziert (Start an Adresse 830). Innerhalb von 100 Durchläufen erreicht unsere Routine einen Vorsprung von $^{18}/_{60}$ Millisekunden gegenüber einem entsprechenden Basic-Programm. Wie Sie an diesem Beispiel erkennen können, ermöglicht es die USR-Funktion, sowohl von Basic aus zugängliche Routinen wie auch Interpreter-interne Operationen aufzurufen und für eigene Zwecke zu nutzen. In Listing 7 haben wir für Sie bereits eine etwas anspruchsvollere Routine vobereitet. Dieses Programm berechnet aus einem beliebigen Variablenfeld (Array) die Summe der einzelnen Argumente. In diesem Fall muß bei Aufruf der USR-Funktion der Namen des Feldes angegeben werden und der USR-Vektor auf Adresse 828 gerichtet sein. Wenn Sie dieses Vorhaben von Basic aus realisieren möchten, müssen Sie schon eine Weile warten. Sie werden bereits bemerkt haben, daß Ihnen zur sinnvollen Nutzung der USR-Funktion nicht nur der Befehlssatz der 65xx-Prozessoren geläufig sein sollte. Der richtige Einsatz der ROM-Routinen ist mindestens genauso wichtig. Zu diesem Zweck ist die Benutzung eines ROM-Listings unumgänglich. Nachfolgend erwartet Sie eine kurze Aufstellung der wichtigsten Routinen. Bitte achten Sie bei der Benutzung der Arithmetik-Sequenzen auf die Einsprungadressen. Teilweise befindet sich zu Beginn der Routinen eine Abfrage auf FAC=0. Diese Abfrage sollte beim Aufruf über den USR-Vektor übergangen werden (Weitere ausführliche Informationen über Rechenoperationen und deren Anwendung finden Sie in dem Artikel »Rechnen in Maschinensprache«).

B849 : FAC = FAC+0.5
B850 : FAC = Konstante-FAC
B853 : FAC = ARG-FAC
B867 : FAC = FAC+ARG
BA28 : FAC = Konstante * FAC
BA2B : FAC = ARG * FAC
BA8C : ARG = Konstante
BAE2 : FAC = FAC * 10
BB0F : FAC = Konstante/FAC
BB12 : FAC = ARG/FAC
BBA2 : FAC = Konstante
BBFC : FAC = ARG
BC0C : ARG = FAC
BF78 : FAC = $ARG^{Konstante}$
BF7B : FAC = ARG^{FAC}

Wie bereits erwähnt, können natürlich alle von Basic aus nutzbaren Routinen ebenfalls verwendet werden (SQR, SIN ...). Abschließend noch eine Anregung. Versuchen Sie einmal, mehrere Basic-Befehle unter der USR-Funktion zusammenzufassen. Auch durch diesen Trick kann man Basic beschleunigen.

SYS und mehr

Gerade haben wir behauptet, die USR-Funktion sei der Weisheit letzter Schluß, und jetzt beschäftigen wir uns doch wieder mit dem SYS-Befehl. Bitte glauben Sie nicht, wir wären inkonsequent. Es soll hier nur noch auf eine weitere Technik der Datenübergabe hingewiesen werden, die übrigens auch im Umgang mit der USR-Funktion verwendet werden kann. Auf einfachste Weise ist es möglich, nicht nur einen Parameter (USR), sondern beliebig viele zu übernehmen. Hierfür bieten sich gleich drei Routinen an:

AE83 : holt das nächste Element eines Ausdrucks in FAC
B79E : holt ein Byte in das X-Register
0073 : holt nächstes Zeichen in Akku

Das Betriebssystem hält sogar noch einige Routinen zur Prüfung der zu übergebenden Parameter bereit. Aber bitte bemühen Sie in diesen Fällen Ihr ROM-Listing. Zum Abschluß finden Sie in Listing 8 eine PRINT-AT-Simulation. Der Einsprung erfolgt über SYS 830, spalte, zeile, »text« (siehe auch »Rechnen in Maschinensprache«). (Erhart/ef)

Sprites ohne Grenzen

Möchten Sie den ganzen Bildschirm für die Darstellung von Sprites und Laufschriften nutzen? Hyperscreen III gibt Ihnen die Gelegenheit dazu. Mit diesem Programm sind Ihrem Bildausschnitt rechts und links keine Grenzen mehr gesetzt. Ein Demo liefern wir gleich mit.

Mit einem genialen Trick hat es der Autor, Hermann Schinagl, geschafft, Sprites und Laufschriften im Rahmen neben dem Textbildschirm erscheinen zu lassen. »Hyperscreen III« (Listing 1) ist das Programm, das dies alles möglich macht.

Verwirklicht wird das alles mit einem kleinen, aber eindrucksvollen Kniff.

Der Trick

Der VIC (Video Interface Controller) liest bei jeder achten Rasterzeile den Zeichengenerator aus, um Buchstaben auf dem Textbildschirm darstellen zu können. Dabei wird auch der Bildschirmrand jedesmal abgeschaltet. Man kann in diesen Prozeß nur durch einen kleinen Kniff eingreifen. Man beeinflußt den Wert, der dem VIC angibt, welches die aktuelle Rasterzeile ist. Wenn dieser Wert nie durch acht teilbar ist, »vergißt« der VIC das Neustellen der Ränder. Mit Hilfe von Hyperscreen III erreicht dieser Wert, der das Stellen der Ränder bewirkt, nie den ominösen Wert acht. So, nun können die Einstellungen für den Rand durch ein Programm geändert werden. Das war's dann. Ein Hinweis noch, wenn Sie diesen Trick selbst benutzen möchten: Die Schleife, die den Rand abstellt, darf nie eine Speicher-Page kreuzen. Das ganze Timing der Routine käme dabei völlig durcheinander.

Sollten nach Veränderung der Scroll-Routine die Sprites leicht anfangen zu flackern, ist die Anpassung der Schleife zu ändern.

Sie können das Programm auch dann nutzen, wenn Sie keine Programmierkenntnisse in Assembler besitzen. Geben Sie Listing 1 mit dem MSE ein und speichern es. Danach geben Sie nun das kleine Basic-Programm in Listing 2 ein und speichern es ebenfalls. Dann starten Sie das Basic-Programm mit »RUN«. Das Programm fordert Sie auf, einen Text einzutippen. Dieser wird nach Drücken von <RETURN> von links nach rechts über den gesamten Bildschirm bewegt. Natürlich können Sie die Routine auch von komplexeren Basic-Programmen aus ansteuern. Nutzen Sie die Möglichkeit, den ganzen Bildschirm nach Ihren Vorstellungen zu gestalten. Vielleicht haben Sie sogar noch Verbesserungen zu der Hauptroutine? Experimente mit dem Programm werden sicher zu überraschenden Ergebnissen führen.

(Hermann Schinagl/kn)

Kurzinfo: Hyperscreen

Programmart: Tool für Sprites
Laden: LOAD "HYPER III.BAS",8
Start: Nach dem Laden RUN eingeben
Besonderheiten: Das Demo-Programm lädt den Maschinensprachteil »HYPER III.OBJ« nach und startet ihn mit SYS 49152. Das Tool läßt sich wie im Demo-Programm leicht in eigene Programme einbinden.
Programmautor: Hermann Schinagl

```
Name : hyper iii .obj     c000 c195
-----------------------------------
c000 : a9 c8 8d 16 d0 a9 00 8d  a9
c008 : 20 d0 8d 21 d0 20 44 e5  03
c010 : a9 08 85 a7 20 41 c0 20  63
c018 : 1d c0 4c 1a c0 78 a9 00  62
c020 : 8d 0e dc a9 7d a2 c0 8d  2c
c028 : 14 03 8e 15 03 a9 95 8d  f3
c030 : 12 d0 a9 01 8d 1a d0 ad  7d
c038 : 11 d0 29 7f 8d 11 d0 58  41
c040 : 60 a2 00 bd 57 c1 9d 00  a3
c048 : d0 e8 e0 11 d0 f5 a9 ff  4a
c050 : 8d 15 d0 8d 1d d0 a9 07  5b
c058 : a8 99 27 d0 88 10 fa a2  eb
c060 : ff a0 07 8a 99 f8 07 ca  d6
c068 : 88 10 f8 a0 00 98 99 00  76
c070 : 3d 99 00 3e 99 00 3f c8  6a
c078 : d0 f4 4c 4e c1 a2 07 ca  82
c080 : d0 fd ea ea a2 16 24 ea  a8
c088 : ce 16 d0 ee 16 d0 ac 12  32
c090 : d0 88 ea 98 29 07 09 18  91
c098 : 8d 11 d0 ea ea ea ea ca  86
c0a0 : 10 e4 a9 1b 8d 11 d0 a9  e8
c0a8 : 01 8d 19 d0 20 b7 c0 a2  d8
c0b0 : f1 ca d0 fd 4c bc fe a2  e6
c0b8 : 00 18 3e 00 3d 3e c8 3f  bb
c0c0 : 3e c7 3f 3e c6 3f 3e 88  ea
c0c8 : 3f 3e 87 3f 3e 86 3f 3e  82
c0d0 : 48 3f 3e 47 3f 3e 46 3f  ae
c0d8 : 3e 08 3f 3e 07 3f 3e 06  21
c0e0 : 3f 3e c8 3e 3e c7 3e 3e  d0
c0e8 : c6 3e 3e 88 3e 3e 87 3e  de
c0f0 : 3e 86 3e 3e 48 3e 3e 47  c7
c0f8 : 3e 3e 46 3e 3e 08 3e 3e  48
c100 : 07 3e 3e 06 3e e8 e8 e8  17
c108 : e0 18 d0 ad c6 a7 d0 46  58
c110 : a0 00 84 fb b1 9e 0a 26  d5
c118 : fb 0a 26 fb 0a 26 fb 85  ee
c120 : fa a5 fb 69 d8 85 fb a2  08
c128 : 00 a0 00 a9 33 85 01 b1  74
c130 : fa 9d 00 3d e8 e8 e8 c8  ac
c138 : c0 08 d0 f3 84 a7 a9 37  49
c140 : 85 01 e6 9e d0 02 e6 9f  cb
c148 : a0 00 b1 9e d0 08 a9 68  ed
c150 : a2 c1 85 9e 86 9f 60 f1  d3
c158 : 97 29 97 59 97 89 97 b9  2c
c160 : 97 e9 97 19 97 49 97 c1  9a
c168 : 48 59 50 45 52 53 43 52  8b
c170 : 45 45 4e 20 49 49 49 20  34
c178 : 57 52 49 54 54 45 4e 20  be
c180 : 42 59 20 48 45 52 4d 41  1f
c188 : 4e 4e 20 53 43 48 49 4e  a8
c190 : 41 47 4c 20 00 8d 41 03  03
```

Listing 1. Hyperscreen III ermöglicht die Darstellung von Sprites zusätzlich zum Textbildschirm

```
100 IF A=0 THEN A=1:LOAD"HYPER III .OBJ",8
,1                                    <035>
105 INPUT"{CLR}IHRE TEXTEINGABE";A$    <062>
110 FOR I=1 TO LEN(A$)                 <156>
115 POKE 49511+I,ASC(MID$(A$,I,1))     <186>
120 NEXT                               <130>
125 POKE 49511+I,32:POKE 49512+I,0     <117>
130 SYS 49152                          <188>
```

Listing 2. Das kurze Basic-Programm fragt nach einem Text, der dann als Laufschrift auf dem Bildschirm ausgegeben wird

Das Schatzkästchen

Wie in jedem Tips & Tricks-Sonderheft haben wir auch diesmal wieder die besten »Kunstgriffe« für Sie gesammelt und drucken sie hier ab. Das Spektrum ist weit: Für jeden Geschmack, ob Einsteiger, »Zweisteiger«, Fortgeschrittener oder Profi, für jeden ist das Richtige dabei. Damit Sie auch schnell finden, was Sie suchen, haben wir die Sammlung in vier Rubriken aufgeteilt (siehe Tabelle). Einfache Grundlagen, wie ein Renew-Befehl oder der Einzeiler zum Zahlenraten fehlen ebensowenig wie »harte Kaliber«, beispielsweise Informationen über die Wirkung des Rasterzeilen-Interrupts.

Hoffen wir, daß diese 34 Tricks Ihnen weiterhelfen, wenn Sie ein Problem haben. Sie sollen aber auch Anreiz sein, selbst auf »Entdeckungsreise« beim C 64 zu gehen. Viel Spaß beim Ausprobieren und Experimentieren.

(Nikolaus Heusler/ef)

Kurzinfo

Themenbereich	Tip-Nummer	ab Seite
a) Grafik, Drucker	1 bis 13	120
b) Programmieren	14 bis 27	136
c) Mathematik	28 bis 32	144
d) C64, Floppy	33 bis 34	146

1. Die Wahrheit über den Rasterzeilen-Interrupt

Ein interessantes Betätigungsfeld für Assembler-Programmierer ist es, die Bilddarstellung durch den Videochip (VIC) aktiv, das heißt bei jedem Bildaufbau aufs Neue, zu beeinflussen. Beispiele dafür sind verschiedene Farben oder Darstellungsweisen in verschiedenen Bildschirmteilen, das Ausschalten des Bildschirmrandes, mehr als acht Sprites gleichzeitig und vieles mehr.

Damit solche Effekte perfekt sind, müssen sie störungsfrei und immer gleich gelingen; jedes Zucken, Flackern, Flimmern und jede Bildstörung stellen einen gravierenden Mangel dar. Doch gerade mit diesen Feinheiten gibt es immer wieder Probleme oder zumindest viel Arbeit und Ärger. Warum dies so ist und doch so sein muß, soll hier untersucht werden.

Für eine makellose Beeinflussung der Bilddarstellung ist es erforderlich, bestimmte Timingbedingungen einzuhalten, das heißt die Manipulationen nur in bestimmten, genau begrenzten Phasen des Bildaufbaus durchzuführen. Dazu ist es notwendig, die momentane Phase des Bildaufbaus mit genügender Genauigkeit festzustellen und den zeitlichen Ablauf des Programms darauf abzustimmen. Die Phasen und damit die Toleranzgrenzen hängen dabei vom gewünschten Effekt ab.

Der VIC liefert verschiedene Rückmeldungen über seine Tätigkeit, und zwar:

- Register 18 ($12) + Bit 7 von Register 17 ($11): die Nummer der im Moment dargestellten Rasterzeile.
- Register 19 ($13) und 20 ($14): die Position in X-Richtung (halbiert) und in Y-Richtung, bei der der Lichtgriffel einen Impuls abgab.
- Register 30 ($1E) und 31 ($1F): Flags für Sprite-Sprite- und Sprite-Hintergrund-Kollisionen.

Das Prüfen dieser Register kann man dem VIC insoweit



Tips & Tricks sind das Salz in der Suppe jedes Programmierers. Doch nur selten findet man aus der Vielzahl der veröffentlichten Tricks auf Anhieb den richtigen heraus. Auf den folgenden Seiten erwartet Sie daher eine geballte Ladung dieser kleinen nützlichen Kniffe.

übertragen, als man ihn programmieren kann, beim
- Erreichen einer gewünschten Rasterzeile,
- Registrieren eines Lichtgriffelimpulses oder
- Feststellen einer Spritekollision mit einem anderen Sprite oder dem Hintergrund eine Interruptanforderung (IRQ) an den Prozessor (die CPU) zu richten. Dadurch kann ein Programm bei Bedarf aufgerufen werden, ohne Rechenzeit auf das Prüfen der Register zu ver(sch)wenden. Die Rückmeldung über solche Ereignisse erfolgt in Register 25 ($19).

Am besten und einfachsten geeignet zur Bestimmung der aktuellen Bildaufbauphase ist die Nummer der momentan dargestellten Rasterzeile. Liest ein Programm diese gelegentlich aus, so erfährt es zwar die Zeilennummer, doch die Kenntnis der Bildaufbauphase ist recht ungenau, da nicht klar ist, welcher Teil innerhalb der Zeile dargestellt wird. Mit zunehmender Häufigkeit der Prüfung der Rasterzeilennummer kann man beim ersten Auftreten eines bestimmten Wertes einen immer engeren Bereich vom Zeilenanfang ab angeben, innerhalb dessen der Bildaufbau gerade stattfindet. Liest man Register 18 ($12) schließlich mit maximaler Häufigkeit, um auf eine bestimmte Zeile zu warten, so beträgt die Unsicherheit darüber, wann die Zeile begonnen wurde, nur noch sechs Taktzyklen (Tz) (der Abstand zweier Prüfungen (7 Tz) minus 1 Tz):

```
      LDA  #zeile
warte CMP  register18 ;4 Tz
      BNE  warte      ;2/3 Tz
```

War die Zeile bei einem Auslesen nicht erreicht, jedoch beim nächsten, so erfolgte der Wechsel 1 bis 7 Tz nach dem ersten Auslesen, die Differenz und damit die Ungenauigkeit in der Kenntnis des Zeitpunkts beträgt also 6 Tz.

Programmiert man den VIC so, daß er einen Raster-IRQ auslöst, so geschieht dies exakt zu Beginn der gewünschten Zeile. Doch die CPU nimmt diese Interruptanforderung, wenn überhaupt, nur nach Abarbeitung ihres momentanen Befehls an. Die Zeitdauer der Maschinenbefehle variiert nun zwischen 2 und 7 (8 bei einigen »illegalen« Opcodes) Taktzyklen, wodurch die IRQ-Annahme um bis zu 6 (7) Tz verzögert sein kann. Diese Verzögerung ist scheinbar zufällig, da nicht ohne weiteres vorhersagbar ist, bei welchem Befehl und in welcher Bearbeitungsphase die Interruptanforderung auftritt. Ich sagte absichtlich »scheinbar«, da man in gewissen Fällen für eine während eines Programmlaufs gleiche, von Programmstart zu Programmstart jedoch variierende Verzögerung sorgen kann. Dazu muß das Hauptprogramm, das unterbrochen werden soll, periodisch sein, das heißt, nach einer bestimmten Zeit müssen wieder dieselben Befehle abgearbeitet werden oder zumindest solche genau gleicher Zeitdauer. Nun muß man die Dauer der Interruptroutine und eventuell auch die Periodendauer des Hauptprogramms so ausgleichen, daß die Zeit für einen Bildaufbau (19656 Tz) minus die Zeit für Interruptroutine ein Vielfaches der Periodendauer des Hauptprogramms ist: $19656 - I = k \cdot P; k \in \mathbb{N}$.

Dann wird während eines Programmlaufs immer der gleiche Befehl im gleichen Stadium unterbrochen, doch kann dies nach dem Neustart des gesamten Programms ein an-

```
0     -; RASTER-MASTER 0.11 31.10.1988
1     -; (C) 1988 BY  T C
2     -;
3     -.GL BILD       = $1B    ;AN
4     -.EQ ZEILE1     = $26
5     -.EQ ZEILE2     = ZEILE1+2
6     -.EQ NORMAL     = 14
7     -;
8     -.GL IRQVECT    = $0314
9     -;
10    -.GL VICCTRL1   = $D011   ;VIC-CONTROL-REG.
11    -.GL RASTER     = $D012
12    -.GL IRQFLAGS   = $D019
13    -.EQ IRQMASKS   = $D01A
14    -.EQ BORDER     = $D020
15    -;
16    -.EQ CIA1ICR    = $DC0D
17    -;
18    -.EQ OLDIRQ     = $EA31
19    -.EQ IRQRET     = $EA7E
20    -;
21    -;***** MAKROS *****
22    -.MA SETIRQ (ZEILE)
23    -             LDA #BILD!O!((ZEILE/2)!A!$80)
24    -             STA VICCTRL1
25    -             LDA #<(ZEILE)
26    -             STA RASTER
27    -             LDA #$FF        ;LOESCHEN
28    -             STA IRQFLAGS
29    -.RT
30    -;
31    -.MA SETVECT (ROUT)
32    -             LDA #<(ROUT)
33    -             LDY #>(ROUT)
34    -             STA IRQVECT
35    -             STY IRQVECT+1
36    -.RT
37    -;
38    -;+++++ STARTADRESSE +++++
39    -;
40    -             .BA $9000
41    -;
42    -             JMP INIT
43    -             JMP AUS
44    -;
45    -;***** VORBEREITUNG *****
46    -INIT         SEI
47    -; TIMER-IRQ VERHINDERN
48    -             LDA #$7F
49    -             STA CIA1ICR
50    -; RASTER-IRQ ERLAUBEN
51    -             LDA #$01
52    -             STA IRQMASKS
53    -;
54    -             ... SETVECT(ROUT)
55    -             ... SETIRQ(ZEILE1)
56    -             CLI
57    -;
58    -             RTS
59    -;
60    -;***** ENDE *****
61    -AUS          SEI
62    -; RASTER-IRQ VERHINDERN
63    -             LDA #$00
64    -             STA IRQMASKS
65    -; TIMER-IRQ ERLAUBEN
66    -             LDA #$81
67    -             STA CIA1ICR
68    -             ... SETVECT(OLDIRQ)
69    -;
70    -             CLI
71    -             RTS
72    -;
73    -;***** IRQ-_OUTINEN *****
74    -;
75    -TOOLATE      JMP IRQRET
76    -;
77    -;----- GRUENER STRICH = 'RASTER-MASTER'
78    -;
79    -ROUT         LDA #<(ZEILE2)
80    -             CMP RASTER
81    -             BCC TOOLATE
82    -             BEQ TOOLATE
83    -;
84    -WZEILE       CMP RASTER
85    -             BNE WZEILE
86    -;
87    -             LDX #10
88    -WARTE1       DEX
89    -             BNE WARTE1
90    -;
91    -             NOP
92    -             LDA RASTER      ;59-66
93    -             CMP #<(ZEILE2+1)
94    -             BEQ OK1
95    -             BIT $AA
96    -             NOP
97    -; 68-71
98    -OK1          LDX #9
99    -WARTE2       DEX
100   -             BNE WARTE2
101   -;
102   -             NOP
103   -             NOP
104   -             NOP
105   -             LDA RASTER      ;124-127
106   -             CMP #<(ZEILE2+2)
107   -             BEQ OK2
108   -             BIT $AA
109   -; 131-132
110   -OK2          LDX #10
111   -WARTE3       DEX
112   -             BNE WARTE3
113   -;
114   -             NOP
115   -             LDA RASTER      ;188-189
116   -             CMP #<(ZEILE2+3)
117   -             BNE OK3
118   -; 194!
119   -;---------
120   -OK3          LDX #3
121   -WARTE4       DEX
122   -             BNE WARTE4
123   -;
124   -             NOP
125   -             NOP
126   -             NOP
127   -             LDA #0
128   -             STA BORDER
129   -             LDA #NORMAL
130   -             STA BORDER
131   -;
132   -             LDA #$FF
133   -             STA IRQFLAGS
134   -             JMP OLDIRQ
135   -;
136   -;+ + + + +  E N D E  + + + + +
```

Listing 1. Das Source-Listing von »Raster Master« im Hypra-Ass-Format

derer Befehl beziehungsweise ein anderes Stadium sein. Durch Ausrechnen oder Ausprobieren kann man so eine Zeitbalance schaffen, so daß zum Beispiel Farbumschaltungen nicht flimmern. Doch diese Balance ist empfindlich und kann durch eine kurzfristige Änderung der Periode oder einen nicht periodischen Ablauf des Hauptprogramms gestört werden. Ein Tastendruck, Bildschirmscrolling oder auch ein anderer C 64 können schon zum Flimmern führen.

Wichtig für eine einigermaßen exakte Reaktion auf IRQs ist auch, daß ihre Annahme nicht vom Hauptprogramm kurzfristig verhindert wird. Gibt es dennoch zwingende Gründe, ein kurzzeitiges Interruptverbot zuzulassen, so muß die Interruptroutine mögliche dadurch entstehende Verzögerungen abpuffern. So kann zum Beispiel auf eine spätere Rasterzeile gewartet werden, mit dem Erfolg, daß der Interrupt auf jeden Fall begonnen wird, ehe diese spätere Rasterzeile erreicht ist. Dazu muß der Abstand zur Zeile, in der der Interrupt ausgelöst wird, natürlich entsprechend der möglichen Dauer des Interruptverbots genügend groß gewählt werden. Im Direktmodus des Basic zum Beispiel ist eine solche Pufferung nötig, da beim Lesen eines Zeichens von der Tastatur Interrupts verboten werden, solange im Tastaturpuffer Zeichen nachgerückt werden.

Für einige wenige Effekte genügt bereits eine grobe Kenntnis der momentanen Bildaufbauphase, dann reicht sogar ein ungepufferter IRQ. Ein Beispiel ist das Ausschalten des senkrechten Randes mit drei Rasterzeilen Toleranz. Umschaltungen an gerade nicht angezeigten Farben oder Darstellungsmodi tolerieren Ungenauigkeiten, diese können mit einem gepufferten IRQ leicht erreicht werden. Für Umschaltungen an gerade sichtbaren Farben oder das Ausschalten des seitlichen Randes benötigt man ein genaues Zeitverhältnis zum Bildaufbau.

Hierzu möchte ich mein Verfahren (Listing 1 und 2) vorstellen, das die Ungenauigkeiten mit jedem Rasterzeilenwechsel halbiert. Zuerst wird ein Interrupt so programmiert, daß die Ungenauigkeit unter 8 Tz liegt. Dann wird solange gewartet, bis bei einem mittleren Wert für die Verzögerung gegenüber dem Idealfall (sofortige Reaktion auf Erreichen

```
Name : raster m.bas        0801 08eb
------------------------------------
0801 : 13 08 40 00 00 11 08 c4  5b
0809 : 07 9e 20 32 31 30 33 20  4f
0811 : 20 00 35 08 c4 07 2a 2a  01
0819 : 2a 20 27 52 41 53 54 45  f2
0821 : 52 20 4d 41 53 54 45 52  90
0829 : 27 20 42 59 20 54 43 20  0e
0831 : 2a 2a 2a 00 00 00 a0 9c  b7
0839 : b9 4c 08 99 ff 8f 88 d0  8e
0841 : f7 a9 34 85 7a a9 17 a0  5d
0849 : 08 20 1e ab 4c 06 90 4c  2e
0851 : 2c 90 78 a9 7f 8d 0d dc  6b
0859 : a9 01 8d 1a d0 a9 46 a0  de
0861 : 90 8d 14 03 8c 15 03 a9  ee
0869 : 1b 8d 11 d0 a9 26 8d 12  cf
0871 : d0 a9 ff 8d 19 d0 58 60  02
0879 : 78 a9 00 8d 1a d0 a9 81  49
0881 : 8d 0d dc a9 31 a0 ea 8d  e0
0889 : 14 03 8c 15 03 58 60 4c  f2
0891 : 7e ea a9 28 cd 12 d0 90  c6
0899 : f6 f0 f4 cd 12 d0 d0 fb  e1
08a1 : a2 0a ca d0 fd ea ad 12  27
08a9 : d0 c9 29 f0 03 24 aa ea  98
08b1 : a2 09 ca d0 fd ea ea ea  5d
08b9 : ad 12 d0 c9 2a f0 02 24  57
08c1 : aa a2 0a ca d0 fd ea ad  9c
08c9 : 12 d0 c9 2b d0 00 a2 03  b9
08d1 : ca d0 fd ea ea ea a9 00  8d
08d9 : 8d 20 d0 a9 0e 8d 20 d0  4f
08e1 : a9 ff 8d 19 d0 4c 31 ea  1a
08e9 : 01 08 0f 4c 23 b1 20 d2  22
```

Listing 2. »Raster Master« geben Sie bitte mit dem MSE ein

der gewünschten Zeile) nun der nächste Zeilenwechsel stattfände. Dies wird geprüft und so entschieden, ob die tatsächliche Verzögerung kurz oder lang ist. Durch entsprechende Korrektur des Zeitverhaltens wird die Ungenauigkeit halbiert. Dazu wird bei kurzer Verzögerung zusätzlich für die Dauer der mittleren Verzögerung gewartet (4 Tz). So werden kurze Verzögerungen (0 bis 3 Tz) und lange (4 bis 7 Tz) zur Überlagerung gebracht (0+4=4, 1+4=5, ..., 3+4=7). Durch drei aufeinanderfolgende, geeignet angepaßte Halbierungen wird eine maximale Ungenauigkeit von 7 Tz korrigiert. Danach ist das Zeitverhältnis des Interruptprogramms zum Bildaufbau eindeutig festgelegt. Zur Verdeutlichung habe ich im Source-Listing (Listing 1) die Zahl der Taktzyklen seit Beginn von »zeile2« eingetragen: Man erkennt die Halbierung der Ungenauigkeit nach jedem Zeilenwechsel. Wird keine absolute Genauigkeit benötigt, kann man Halbierungen einsparen.

Ich nannte anfänglich drei Rückmeldungen des VIC über den Bildaufbau. Man könnte sie alle zur Feststellung der momentanen Bildaufbauphase verwenden. Die Rasterzeilennummer ist jedoch die einfachste Möglichkeit, da sie direkt lesbar ist. Ein Auslösen des Lichtgriffelimpulses, wie A. Beermann es tat, erfordert einige Programmiertricks und kann durch Impulse von außen auf dieser Leitung gestört werden (Leertaste, Joystick-Feuerknopf in Port 1, Lichtgriffel).

Eher theoretisch ist wohl die Möglichkeit, durch eigens dafür gezielt plazierte Spritekollisionen die Bildaufbauphase zu erkennen. (Thomas Chadzelek/ef)

2. Das Geheimnis von $3FFF

Die Speicherzeile $3FFF ist trotz ihrer Funktion wenig bekannt. Schaltet man mittels eines Raster-Interrupts den unteren Rand ab, so muß in ihr der Wert Null stehen, damit im Bereich des ehemaligen Randes keine schwarzen Streifen stören.

Denn das Bitmuster dieser Adresse wird nämlich 40mal pro Rasterzeile auf den Bildschirm geschrieben. Ändert man dieses Bitmuster in jeder Rasterzeile, so kann man interessante Muster auf dem Rand erzeugen. Und das alles völlig ohne Sprites! Listing 3 demonstriert diesen hochinteressanten Effekt.

```
10 POKE 53280,0:POKE 53281,0:PRINT"{CLR,3D
   OWN,WHITE,4SPACE}RANDDEMO VON S.GOEBBEL
   S (C) 1987"                             <234>
15 PRINT TAB(13)"{DOWN,GREY 1}BITTE WARTEN
   ...{DOWN}"                              <196>
20 FOR I=0 TO 209:READ Q:POKE 36864+I,Q:NE
   XT                                      <091>
30 FOR I=0 TO 3:REM ZEICHEN DEFINIEREN     <128>
40 FOR A=0 TO 7                            <045>
50 POKE 37120+A+I*8,2↑A                    <171>
51 POKE 37144+A+I*8,2↑(7-A):               <253>
60 NEXT A,I                                <059>
70 FOR I=0 TO 48 STEP 8:REM FARBEN SETZEN  <130>
80 POKE 37168+I,1:POKE 37169+I,3:POKE 3717
   0+I,5:POKE 37171+I,13                   <186>
85 POKE 37172+I,13:POKE 37173+I,5:POKE 371
   74+I,3:POKE 37175+I,1                   <006>
90 NEXT                                    <100>
100 SYS 36864:REM MASCINENPROGRAMM :       <201>
32000 DATA 120,169,31,141,20,3,169,144,141
      ,21,3,173,17,208,41,127,141,17,208,1
      69                                   <243>
32001 DATA 186,141,18,208,169,129,141,26,2
      08,88,96,173,25,208,141,25,208,48,7  <063>
32002 DATA 173,13,220,88,76,49,234,173,254
      ,144,201,6,240,18,169,6,141,254,144  <171>
32003 DATA 169,1,141,18,208,169,19,141,17,
      208,76,107,144,169,0,141,254,144,169 <214>
32004 DATA 248,141,18,208,169,27,141,17,20
      8,76,157,144,120,169,49,141,20,3,169 <243>
32005 DATA 234,141,21,3,169,240,141,26,208
      ,88,96,173,0,145,141,255,63,160,0,23
      4                                    <117>
32006 DATA 234,234,162,0,232,224,8,208,251
      ,185,1,145,141,255,63,185,48,145,141 <108>
32007 DATA 33,208,162,0,232,224,18,208,251
      ,200,192,47,208,232,169,0,141,33,208 <207>
32008 DATA 76,188,254,173,0,145,141,255,14
      5,162,0,189,1,145,157,0,145,232,224  <165>
32009 DATA 47,208,245,173,255,145,141,47,1
      45,173,95,145,141,255,145,162,63,189 <072>
32010 DATA 47,145,157,48,145,202,224,255,2
      08,245,173,255,145,141,48,145,76,188 <197>
32011 DATA 254                             <125>
33000 PRINT"{6DOWN,CYAN}DIESES PROGRAMM KO
      MMT OHNE (!) SPRITES"                <169>
33010 PRINT"AUS. DIE ZEICHEN AUF DEN RAEND
      ERN WER-"                            <236>
33020 PRINT"DEN DURCH AENDERUNG DES WERTES
       IN $3FFF"                           <244>
33030 PRINT"ERZEUGT."                      <029>
33035 PRINT"{DOWN}SIE FINDEN DAS MASCHINEN
      PRG. AB $9000"                       <179>
33040 PRINT TAB(13)"{WHITE,DOWN}<TASTENDRU
      CK>":POKE 198,0:WAIT 198,1:POKE 198,
      0                                    <070>
33050 SYS 36953:POKE 53281,0:GOSUB 35000   <242>
33060 FOR I=0 TO 7:READ Q:POKE 37120+I,Q:P
      OKE 37144+I,Q:POKE 37128+I,Q:POKE 37
      136+I,Q                              <128>
33070 POKE 37152+I,Q:POKE 37160+I,Q:NEXT   <087>
33080 DATA 129,195,231,255,255,231,195,129 <107>
33090 FOR I=0 TO 47:POKE 37168+I,11:NEXT   <092>
34100 SYS 36864:POKE 198,0:WAIT 198,1:POKE
       198,0:SYS 36953                     <231>
34110 FOR I=0 TO 48 STEP 8                 <120>
34120 POKE 37168+I,14:POKE 37169+I,11:POKE
       37170+I,12:POKE 37171+I,15          <217>
34130 POKE 37172+I,15:POKE 37173+I,12:POKE
       37174+I,11:POKE 37175+I,14          <022>
34140 NEXT:SYS 36864                       <216>
34150 POKE 198,0:WAIT 198,1:POKE 198,0:SYS
       36953:RESTORE:GOTO 10               <070>
35000 PRINT"{CLR,WHITE}AUFBAU DES MASCHINE
      NPROGRAMMS:"                         <253>
35010 PRINT"{2DOWN,CYAN}DER ERSTE RASTERIN
      TERRUPT BEI $0F8 SORGT"              <060>
35020 PRINT"{UP,WHITE}FUER DAS AUSSCHALTEN
       DES RANDES, INDEM"                  <064>
35030 PRINT"{CYAN}AUF 24 ZEILEN UMGESCHALT
      ET WIRD. HIER"                       <179>
35040 PRINT"{WHITE}BEGINNT DANN EINE SCHLE
      IFE, DIE DIE VER-"                   <216>
35050 PRINT"{CYAN,UP}SCHIEDENEN WERTE IN D
      AS REGISTER $3FFF"                   <139>
35060 PRINT"{WHITE}SCHREIBT. ZWISCHEN DIES
      EN AENDERUNGEN"                      <235>
35070 PRINT"{CYAN}WIRD EINE WARTESCHLEIFE
      DURCHLAUFEN,DIE"                     <108>
35080 PRINT"{WHITE}DANN BEENDET IST, WENN
      DER ELEKTRONEN-"                     <094>
35090 PRINT"{CYAN}STRAHL EINE NEUE ZEILE E
      RREICHT."                            <076>
35100 PRINT"{DOWN,WHITE}EIN ZWEITER INTERR
      UPT SCHALTET DANACH "                <072>
35110 PRINT"{CYAN}WIEDER AUF 25 ZEILEN UM.
      "                                    <224>
35120 PRINT"{DOWN,WHITE}DIE ZEICHENDATEN L
      IEGEN AB $9100, DIE"                 <151>
35130 PRINT"{CYAN}FARBDATEN AB $9130. BEID
      E FELDER WER-"                       <079>
35140 PRINT"{WHITE}GESCROLLT, SO DASS DIE
      ERZEUGTE GRAFIK"                     <198>
35150 PRINT"{CYAN}BEWEGT WIRD."            <027>
35160 PRINT"{WHITE,2DOWN}DAS MASCHINENPROG
      RAMM WIRD MIT SYS36864"              <159>
35170 PRINT"{CYAN}GESTARTET UND MIT SYS369
      53 BEENDET.":RETURN                  <073>
35180 REM STEFFEN GOEBBELS                 <048>
35190 REM ALTE HEERSTR. 25                 <174>
35200 REM 4179 WEEZE     1                 <217>
```

Listing 3. »rand« demonstriert die Verwendung von $3FFF

Funktion des Maschinenprogramms

Zuerst werden alle Vorbereitungen zur Benutzung von Raster-Interrupts getroffen. Mit ihnen wird der oberen und untere Bildschirmrand ausgeschaltet. Wie dies geschieht, wurde schon mehrmals im 64'er-Magazin geschrieben (z.B. 64'er, 5/87, Seite 47).

Erreicht der Rasterstrahl den unteren Rand, so gelangt das Programm in eine Schleife, die so lange andauert, bis der Strahl wieder den beschreibbaren Teil des Bildschirms erreicht. Während dieser Schleife werden Daten in die Speicherzellen $3FFF und $D021 geschrieben. Dieser

Schreibvorgang wird von einer Verzögerungsschleife so gesteuert, daß er jeweils bei Erreichen einer neuen Rasterzeile stattfindet. Mit $D021 werden die Hintergrundfarben geändert, mit $3FFF das Bildmuster. Denn diese Speicherstelle ist für das Aussehen des Randes verantwortlich: Ihre Bits werden in jeder Rasterzeile 40mal nebeneinander dargestellt. In einer solchen Zeile läßt sich dieser Wert aus Zeitgründen nicht ändern. Aber jede Zeile kann von einem neuen Wert bestimmt sein, so daß interessante Muster erzielt werden können. Auf dem Rand lassen sich also Zeichen darstellen, wobei jedes Zeichen 40mal in einer Zeile steht.

Ein weiterer Teil des Programms bewegt die Datenfelder, so daß die Randinformation gescrollt wird.

Das gesamte Programm »hängt« im Interrupt des C64, so daß bei Verlassen des Basic-Programms durch Drücken der STOP-Taste der Rand weiterhin verändert wird. Die beschriebene Funktion bezieht sich übrigens nicht nur auf die Adresse $3FFF, sondern generell immer auf die letzte Adresse des 16 K-Blocks, den der VIC gerade ansprechen kann. Also $3FFF, $7FFF, $BFFF oder $FFFF.

(Steffen Goebbels/André Moll/ef)

3. Kopfzeilen per Raster-Interrupt

Dieses Programm (Listing 4) bietet dem Programmierer in Basic oder Maschinensprache bis zu drei Statuszeilen am oberen Rand des Bildschirms.

Besonderheiten:

»Kopfzeilen« arbeitet mit Rasterzeilen-Interrupt. Dadurch ist es nicht nötig, das Betriebssystem ins RAM zu kopieren, wie es die üblichen Statuszeilen-Programme machen. So bleibt das RAM unter dem ROM frei und kann für sinnvolle-re Zwecke verwendet werden.

Das Programm ist in Maschinensprache geschrieben und belegt den Speicher von 49152 bis 49301 ($C000 bis $C095). Weiterhin werden noch 240 Byte als Speicher benötigt.

Bei der Arbeit mit »Kopfzeilen« geht man folgendermaßen vor:

1. Durch »POKE 49152 + 9, ZL« stellen Sie die Anzahl der Statuszeilen ein (ZL: 1 bis 3). Falsche Angaben werden nicht überprüft.
2. Durch »SYS 49152+6« werden die oberen Zeilen (je nach ZL eine bis drei) in den Zwischenspeicher gerettet, auch die Farbbytes.

```
Name : kopfzeilen              c000 c095
----------------------------------------
c000 : 4c 0a c0 4c 62 c0 4c 74   51
c008 : c0 02 78 a9 2d a2 c0 8d   23
c010 : 14 03 8e 15 03 ae 09 c0   37
c018 : bd 90 c0 8d 12 d0 ad 11   80
c020 : d0 29 7f 8d 11 d0 a9 81   58
c028 : 8d 1a d0 58 60 ad 19 d0   7b
c030 : 8d 19 d0 30 07 ad 0d dc   50
c038 : 58 4c 31 ea 78 ae 09 c0   03
c040 : bd 8a c0 aa bd 94 c0 9d   86
c048 : 00 04 bd 0c c1 9d 00 d8   f6
c050 : ca 10 f1 ae 09 c0 ad 12   e6
c058 : d0 dd 8d c0 90 f8 58 4c   5d
c060 : 81 ea 78 a9 00 8d 1a d0   20
c068 : a9 31 a2 ea 8d 14 03 8e   52
c070 : 15 03 58 60 ae 09 c0 bd   db
c078 : 8a c0 aa bd 00 04 9d 94   84
c080 : c0 bd 00 d8 9d 0c c1 ca   11
c088 : 10 f1 60 27 4f 77 3b 43   b2
c090 : 4b 1e 0a 00 00 ff 00 00   6d
```

Listing 4. »Kopfzeilen« per Rasterzeilen-Interrupt

```
10 REM DEMO FUER 'KOPFZEILEN'                    <227>
20 REM                                           <082>
30 REM (W) BERND SCHULLER 1987                   <201>
40 REM                                           <102>
50 IF A=0 THEN A=1:LOAD"KOPFZEILEN",8,1          <128>
60 IN=49152:OF=49152+3                           <179>
70 GT=49152+6:ZL=49152+9                         <195>
80 POKE ZL,3                                     <149>
90 POKE 53280,0:POKE 53281,0:SYS OF              <014>
100 PRINT"«CLR,LIG.BLUE»DEMO FUER DAS PROG
    RAMM«SPACE,RED»'KOPFZEILEN'"                 <058>
110 PRINT"«LIG.BLUE»BIS ZU«SPACE,YELLOW»DR
    EI«LIG.BLUE,SPACE»STATUSZEILEN MOEGLIC
    H !"                                         <018>
120 FOR T=0 TO 39:PRINT"*";:NEXT                 <138>
130 SYS GT:SYS IN                                <091>
140 POKE 198,0                                   <048>
150 PRINT"«DOWN,RVSON»A)«RVOFF,SPACE»DURCH
     RASTER-IRQ KEINE SPEICHER-«5SPACE»PLA
    TZVERSCHWENDUNG"                             <103>
160 GOSUB 230                                    <160>
170 PRINT"«RVSON»B)«RVOFF,SPACE»BETRIEBSSY
    STEM KOPIEREN UNNOETIG !!"                   <230>
180 GOSUB 230                                    <180>
190 PRINT"«RVSON»C)«RVOFF,SPACE»1-3 STATUS
    ZEILEN,«2SPACE»FLACKERFREI"                  <011>
200 GOSUB 230                                    <200>
210 IF PEEK(198)=0 THEN 150                      <120>
220 SYS OF:END                                   <155>
230 FOR T=0 TO 100:NEXT:RETURN                   <128>
```

Listing 5. Demonstration der Fähigkeiten von Listing 4

3. Durch »SYS 49152« wird das Programm gestartet. Ab jetzt wird bei jedem Durchlauf des Elektronenstrahls der Inhalt des Zwischenspeichers in den Bildschirm- und Farbspeicher geschrieben.
4. Mit Hilfe von »SYS 49152+3« können Sie das Programm wieder ausschalten.

Das Demoprogramm in Listing 5 zeigt die Möglichkeiten von »Kopfzeilen«.

(Bernd Schuller/ef)

4. Der Mülleimer

Dieses Programm (Listing 6) wird mit dem Befehl LOAD »MUELLEIMER«,8 geladen und mit RUN gestartet. Sie können nun etwas programmieren, kurze Basic-Programme laden, die den Bereich ab $C000 nicht belegen oder sonst etwas Sinnvolles anstellen. Malen Sie doch einmal eine schöne Blockgrafik auf dem Bildschirm! Nach etwa eineinhalb Minuten passiert's dann: Auf dem Bildschirm erscheint ein Mülleimer, der nach weiteren drei Sekunden aktiv wird und allmählich den gesamten Bildschirm »aufsaugt«. Kurz danach verschwindet er wieder, der C 64 hat sich erholt.

Sie ahnen es sicher schon: In Abständen von etwa weiteren zwei Minuten 50 Sekunden wiederholt sich diese Prozedur.

Nun noch einige technische Hinweise zum Programm: Es läuft im Interrupt, um gleichzeitig mit anderen Routinen aktiv sein zu können. Da es den IRQ-Vektor jedoch unter Berücksichtigung des alten Wertes verändert, kann es zusammen mit anderen IRQ-Programmen ablaufen, die nicht den $C-Bereich »berühren«. Dazu aktivieren Sie zuerst das andere Programm, danach den Mülleimer.

Das Programm »Mülleimer« darf nicht mit ⟨RESTORE⟩ unterbrochen werden, während der Mülleimer »saugt«, da sonst einige für das Betriebssystem wichtige Zellen nicht rekonstruiert werden (u.a. Basic-Pointer 43 bis 46). Der Mülleimer ist, das liegt in der Natur der Sache, nur im Lores-Modus des C 64 lauffähig, berücksichtigt aber die Startadresse des Bildschirms (normal 1024).

Das Programm arbeitet mit Sprites. Die Spritedaten lie-

```
Name : muelleimer           0801 0a84
-------------------------------------
0801 : 17 08 c2 07 9e 20 32 30   c2
0809 : 37 33 2c 4d 55 45 4c 4c   d8
0811 : 45 49 4d 45 52 00 00 00   1c
0819 : 20 f1 b7 a9 50 a2 08 85   9a
0821 : f7 86 f8 a0 00 a9 00 a2   40
0829 : c0 85 f9 86 fa a5 fa c9   57
0831 : c2 d0 09 a5 f9 c9 35 d0   b7
0839 : 03 4c 00 c0 b1 f7 91 f9   8f
0841 : e6 f7 d0 02 e6 f8 e6 f9   5d
0849 : d0 02 e6 fa 18 90 de 78   a6
0851 : ad 15 03 c9 c0 f0 28 8d   d2
0859 : 3a c0 ad 14 03 8d 39 c0   e4
0861 : a9 32 a0 c0 8d 14 03 8c   02
0869 : 15 03 a9 14 8d 81 c0 a9   28
0871 : 00 8d 82 c0 38 6e 7f c0   67
0879 : a9 3b a0 c0 20 1e ab 58   52
0881 : 60 20 83 c0 20 dc c0 4c   6f
0889 : 00 00 0d 57 48 41 54 45   22
0891 : 56 45 52 20 59 4f 55 27   d6
0899 : 4c 4c 20 53 45 45 3a 20   26
08a1 : 44 4f 4e 27 54 20 50 41   0f
08a9 : 4e 49 43 2e 20 45 56 45   43
08b1 : 52 59 2d 54 48 49 4e 47   1c
08b9 : 20 49 53 20 55 4e 44 45   ba
08c1 : 52 20 43 4f 4e 54 52 4f   4e
08c9 : 4c 2e 11 00 00 00 00 00   71
08d1 : 00 00 ad 82 c0 f0 14 ce   0e
08d9 : 80 c0 d0 4e a9 2c 8d 7f   e8
08e1 : c0 a9 28 8d 81 c0 a9 00   f6
08e9 : 8d 82 c0 ce 80 c0 d0 3a   87
08f1 : ce 81 c0 d0 35 ee 82 c0   20
08f9 : a9 01 8d 15 d0 a9 00 8d   9e
0901 : 17 d0 8d 1d d0 8d 1c d0   13
0909 : 8d 10 d0 a9 b5 8d 00 d0   71
0911 : a9 e5 8d 01 d0 a9 01 8d   aa
0919 : 27 d0 a2 3f bd f5 c1 9d   07
0921 : c0 02 ca 10 f7 a9 0b 8d   ab
0929 : f8 07 60 ad 7f c0 30 46   be
0931 : a5 d1 48 a5 d2 48 a5 f3   73
0939 : 48 a5 f4 48 a5 2b 48 a5   ba
0941 : 2c 48 a5 2d 48 a5 2e 48   9b
0949 : ce 7f c0 20 28 c1 20 89   2f
0951 : c1 20 e0 c1 ad 7f c0 10   8d
0959 : ef 68 85 2e 68 85 2d 68   dc
0961 : 85 2c 68 85 2b 68 85 f4   bd
0969 : 68 85 f3 68 85 d2 68 85   39
0971 : d1 a9 00 8d 15 d0 60 a9   75
0979 : d4 85 d1 ad 88 02 18 69   06
0981 : 03 85 d2 20 24 ea a0 00   1c
0989 : b1 d1 8d 7d c0 c8 b1 d1   f3
0991 : 8d 7e c0 a2 17 a5 d1 38   38
0999 : e9 28 85 2b a5 d2 e9 00   f6
09a1 : 85 2c a5 f3 38 e9 28 85   a3
09a9 : 2d a5 f4 e9 00 85 2e a0   49
09b1 : 01 b1 2b 91 d1 b1 2d 91   0a
09b9 : f3 88 10 f5 a5 2b 85 d1   20
09c1 : a5 2c 85 d2 a5 2d 85 f3   fa
09c9 : a5 2e 85 f4 ca 10 c6 a9   21
09d1 : 20 c8 91 d1 c8 91 d1 60   15
09d9 : a9 00 85 d1 ad 88 02 85   50
09e1 : d2 a2 18 20 24 ea a0 14   53
09e9 : b1 d1 c9 20 d0 13 88 b1   24
09f1 : d1 c8 91 d1 88 b1 f3 c8   3c
09f9 : 91 f3 88 d0 f1 a9 20 91   d0
0a01 : d1 a0 15 b1 d1 c9 20 d0   2b
0a09 : 15 c8 b1 d1 88 91 d1 c8   17
0a11 : b1 f3 88 91 f3 c8 c0 27   e7
0a19 : 90 ef a9 20 91 d1 a5 d1   f1
0a21 : 18 69 28 85 d1 a5 d2 69   11
0a29 : 00 85 d2 ca 10 b5 60 ad   85
0a31 : 7d c0 c9 20 d0 08 ad 7e   86
0a39 : c0 c9 20 d0 01 60 a9 2c   12
0a41 : 8d 7f c0 60 03 ff 80 03   02
0a49 : 01 80 3f ff f8 60 00 0c   05
0a51 : 3f ff f8 08 00 20 08 00   f0
0a59 : 20 0b 6d a0 09 24 a0 09   b5
0a61 : 24 a0 09 24 a0 09 24 a0   c0
0a69 : 09 24 a0 09 24 a0 09 24   81
0a71 : a0 09 24 a0 09 24 a0 09   f9
0a79 : 24 a0 0b 6d a0 08 00 20   e8
0a81 : 0f ff e0 e6 fc 60 a5 fb   06
```

Listing 6. Viel Vergnügen mit dem gierigen Mülleimer

gen direkt hinter dem Maschinenprogramm im Speicher und werden jedesmal nach 704 kopiert. Falls der Scherzartikel also unter einem anderen Programm laufen soll, muß dieses folgenden Bedingungen genügen:
- möglichst keine Sprites,
- kein weißer Bildschirm (Mülleimer),
- VIC-Bank = 0,
- Speicherbereich $C000 bis $C200 nicht belegt,
- Speicherbereich 704 bis 767 nicht belegt,
- falls IRQ-Vektor verändert wird, muß der Scherzartikel zuletzt aktiviert werden.

Wir wünschen Ihnen viel Spaß mit dem Programm, das als kleiner Gag sicher jedem gefallen wird.

(Nikolaus Heusler/ef)

5. Der Sprite-Dreher

Bei dem Maschinenprogramm »Drehe Sprites« (Listing 7) für den C 64 handelt es sich um ein Utility, das es gestattet, ein an beliebiger Stelle im Speicher liegendes Sprite um einen beliebigen Winkel entgegen dem Uhrzeigersinn zu drehen und dann an einer ebenfalls beliebigen Stelle im Speicher wieder abzulegen. Die Erstellung von Animationen, zum Beispiel für Spiele, die sich drehende Sprites verwenden, wird durch dieses Hilfsprogramm stark vereinfacht, da außer der Erstellung des Quelle-Sprites und kleinen Schönheitskorrekturen bei den Sprite-Sequenzen keinerlei Arbeit mehr geleistet werden muß. Das Maschinenprogramm dreht ein Sprite um einen beliebigen Winkel. Der Mittelpunkt der Drehung liegt (bei einem in der linken oberen Ecke liegenden Ursprung) bei P (12/10) innerhalb des Sprites. Die Syntax zum Aufruf des Programms lautet:

```
SYS 49358, Adresse des Quellsprites,
           Adresse des Ziel-Sprites,
           Winkel im Bogenmaß
```

Das Programm liegt im Bereich von $C000/$C203. Während des Drehens wird auch noch der Bereich von $C204/$C2FE benötigt. Natürlich kann das Programm auch von Maschinensprache aus aufgerufen werden. Die Adresse des Quellsprites steht in $FB/$FC, die des Zielsprites in $FD/$FE, der Winkel steht im FAC (Fließkomma-Akkumulator). Nachdem diese Werte gesetzt sind, kann die Routine mittels JSR $COEE aufgerufen werden.

(Christian Pdemeyer/ef)

```
Name : drehe sprites        c000 c208
-------------------------------------
c000 : a2 f9 a0 c2 20 d4 bb 20   f7
c008 : 64 e2 a2 f4 a0 c2 20 d4   6f
c010 : bb 20 0c bc a9 0c 20 3c   6a
c018 : bc 20 30 ba 20 b4 bf a9   42
c020 : 00 85 02 a9 04 a2 c2 8d   14
c028 : 40 03 8e 41 03 ae 40 03   62
c030 : ac 41 03 20 d4 bb 18 ad   29
c038 : 40 03 69 05 8d 40 03 90   fd
c040 : 03 ee 41 03 a9 f4 a0 c2   b5
c048 : 20 67 b8 e6 02 a5 02 c9   10
c050 : 18 d0 da a9 f9 a0 c2 20   ac
c058 : a2 bb 20 6b e2 a2 f9 a0   ba
c060 : c2 20 d4 bb 20 0c bc a9   88
c068 : 0c 20 3c bc 20 30 ba 20   da
c070 : b4 bf a9 00 85 02 a9 7c   76
c078 : a2 c2 8d 40 03 8e 41 03   96
c080 : ae 40 03 ac 41 03 20 d4   fb
c088 : bb 18 ad 40 03 69 05 8d   6d
c090 : 40 03 90 03 ee 41 03 a9   2f
c098 : f9 a0 c2 20 67 b8 e6 02   72
c0a0 : a5 02 c9 18 d0 da 60 98   52
c0a8 : 8d 42 03 0a 18 6d 42 03   54
c0b0 : 8d 42 03 8a 38 e9 08 ee   41
c0b8 : 42 03 b0 f9 ce 42 03 69   c5
c0c0 : 08 aa ac 42 03 60 80 40   46
c0c8 : 20 10 08 04 02 01 20 fd   18
c0d0 : ae 20 8a ad 20 f7 b7 84   90
c0d8 : fb 85 fc 20 fd ae 20 8a   c4
c0e0 : ad 20 f7 b7 84 fd 85 fe   de
c0e8 : 20 fd ae 20 8a ad 20 00   4d
c0f0 : c0 a9 00 a0 3f 91 fd 88   22
c0f8 : 10 fb 8d 43 03 8d 44 03   85
c100 : a9 16 8d 47 03 ae 43 03   b9
```

Listing 7. Damit drehen Sie Sprites in jedem beliebigen Winkel

```
c108 : ac 44 03 20 a7 c0 b1 fb  da
c110 : 3d c6 c0 c9 00 d0 03 4c  45
c118 : dc c1 a9 c2 8d 41 03 ad  e2
c120 : 47 03 0a 0a 18 6d 47 03  bd
c128 : 69 7c 8d 40 03 90 03 ee  d9
c130 : 41 03 ad 40 03 ac 41 03  07
c138 : 20 a2 bb a9 c2 8d 41 03  71
c140 : ad 43 03 0a 0a 18 6d 43  2f
c148 : 03 69 04 8d 40 03 90 03  17
c150 : ee 41 03 ad 40 03 ac 41  a7
c158 : 03 20 50 b8 a9 0c 20 7e  0f
c160 : bd 20 49 b8 a5 66 30 74  ce
c168 : 20 f7 b7 c0 18 b0 6d 8c  60
c170 : 45 03 a9 c2 8d 41 03 ad  44
c178 : 47 03 0a 0a 18 6d 47 03  15
c180 : 69 04 8d 40 03 90 03 ee  f5
c188 : 41 03 ad 40 03 ac 41 03  5f
c190 : 20 a2 bb a9 c2 8d 41 03  c9
c198 : ad 43 03 0a 0a 18 6d 43  87
c1a0 : 03 69 7c 8d 40 03 90 03  8d
c1a8 : ee 41 03 ad 40 03 ac 41  ff
c1b0 : 03 20 67 b8 a9 ff a0 c1  55
c1b8 : 20 50 b8 20 49 b8 a5 66  f0
c1c0 : 30 1a 20 f7 b7 c0 15 b0  3c
c1c8 : 13 8c 46 03 ae 45 03 ac  8e
c1d0 : 46 03 20 a7 c0 b1 fd 1d  61
c1d8 : c6 c0 91 fd ee 43 03 ad  93
c1e0 : 43 03 c9 18 f0 03 4c 05  7d
c1e8 : c1 a9 00 8d 43 03 ee 44  c0
c1f0 : 03 ce 47 03 ad 44 03 c9  29
c1f8 : 15 f0 03 4c 05 c1 60 84  b9
c200 : 20 00 00 00 00 00 00 00  21
```

Listing 7. (Schluß)

6. Screenmanager

Es existiert bereits eine große Zahl von Programmen, die den Bildschirm des C 64 in verschiedenfarbige Bereiche einteilen können. Sie haben jedoch meistens den Nachteil, daß Lage, Größe und Anzahl der Farbunterteilungen nicht genügend variabel sind. Außerdem wird meistens der gesamte Bildschirm aufgeteilt. Es ist also nicht möglich, zum Beispiel nur eine Schriftzeile farbig zu unterlegen. Aus diesem Mangel heraus entstand das Programm »Screenmanager« (Listing 10). Es ist eine kurze Maschinenroutine, die im Interrupt des C 64 läuft. Auf der Diskette wird nur ein Block benötigt, man kann sie ohne viel Aufwand von einem Hauptprogramm aus nachladen (mit »,8,1«). Die Routine versetzt auch den Basic-Programmierer ohne Assembler-Kenntnisse in die Lage, verschiedenfarbige Bereiche beliebiger Größe und Anzahl zusätzlich zum normalen Bildschirmhintergrund einzurichten. Definiert werden die Zonen durch eine Liste, die ab Adresse 49300 im Speicher abgelegt wird, und anschließendem Aufruf des Programms mit einem SYS-Befehl. Die Definition einer solchen Liste sieht folgendermaßen aus: Als erstes wird die Rasterzeile-1 angegeben, in der die erste Zone beginnen soll (sinnvoll sind nur Zahlen von 49 bis ca. 250, da außerhalb dieses Bereichs der Bildschirmrahmen liegt). Ab dieser Rasterzeile werden die verschiedenen Bereiche angelegt. Dazu wird nun die Farbe (0 bis 15) des ersten Bereichs angegeben, anschließend seine »Breite« (wieder in Rasterzeilen, beliebig von 1 bis zirka 250, wenn der gesamte Bildschirm umgefärbt werden soll). Nach diesem Schema können Sie jetzt beliebig viele Bereiche anhängen: Farbe des Bereichs, Breite nächste Farbe und so weiter. Zum Beenden dieser Folge wird an die Liste eine beliebige Zahl von 128 bis 255 angehängt. Dahinter können Sie jetzt den Beginn einer neuen Farbzone nach dem gleichen Muster durch Angabe der neuen Startzeile, Farbe, Breite, und so weiter kennzeichnen oder die Liste mit einer Null beenden. Zur Verdeutlichung ein Beispiel: Es sollen ab Zeile 100 und 150 zwei mehrteilige Farbzonen definiert werden. Im Speicher sieht das dann so aus:

49300	99	(Beginn der 1. Zone-1)
49301	2	(1. Farbe = rot)
49302	10	(10 Zeilen breit)
49303	5	(2. Farbe = grün)
49304	10	(10 Zeilen breit)
49305	6	(3. Farbe = blau)
49306	8	(8 Zeilen breit)
49307	255	(Endekennung der 1. Zone)
49308	149	(Beginn der 2. Zone-1)
49309	7	(1. Farbe = gelb)
...		(und so weiter, wie oben)
49314	5	(letzter Bereich 5 Zeilen breit)
49315	255	(Endekennung der Zone)
49316	0	(Ende der Liste)

Nach dem Start mit SYS 49152 läuft die Routine im Raster-IRQ im Hintergrund eines beliebigen Basic-Programms (es ist auch ein Aufruf von Assembler aus möglich, solange keine Interrupts verwendet werden). Die normale Hintergrundfarbe müssen Sie jetzt statt in Adresse 53281 in 49266 ablegen! Vom Programm wird in der Zeropage die Adresse $FB (251) belegt. Das Hauptprogramm wird um so mehr gebremst, je größer die definierten Zonen sind. Durch

```
20   -;*************************
30   -;*** - screenmanager -  ***
40   -;*** 9.+13.11.87 by     ***
41   -;*** k.kaehler          ***
42   -;*** skalitzer str.134a ***
43   -;*** 1000 berlin 36     ***
50   -;*************************
60   -;
70   -.ba $c000
80   -;
90   -;
100  -.eq vi=$d000
101  -.eq rmsb=vi+$11
102  -.eq rline=vi+$12
103  -.eq irqreq=vi+$19
104  -.eq irqmsk=vi+$1a
105  -.eq back=vi+$21
106  -.eq icr=$dc0d
107  -.eq sysirq=$ea31
108  -.eq irqout=$ea81
190  -.eq irqvec=$0314
220  -.eq tabpnt=$fb
230  -;
240  -;***   routine anschalten   ***
250  -;
260  -start     lda #$00 -    ;pointer auf tabellenstart
270  -          sta tabpnt
280  -;
290  -;   irq initialisieren
300  -;
310  -          sei           ;irq sperren
320  -          lda rmsb      ;rasterzeilen-msb:=0
330  -          and #$7f
340  -          sta rmsb
350  -          lda #$7f      ;cia-irqs verhindern
360  -          sta icr
370  -          lda #$01      ;rasterirqs freigeben
380  -          sta irqmsk
390  -          lda #((irq)   ;irq-vektor auf eigene routine
400  -          sta irqvec
410  -          lda #)(irq)
420  -          sta irqvec+1
430  -;
440  -          jsr readtab   ;1.zonenbeginn auslesen
450  -          sta rline     ;=ausloesende zeile
460  -          cli           ;irqs freigeben
470  -          rts           ;ruecksprung ins basic
480  -;
500  -;***   routine abschalten   ***
510  -;
520  -off       sei           ;irqs sperren
530  -          lda #$00      ;raster-irq aus
540  -          sta irqmsk
550  -          lda #$81      ;cia-irqs ermoeglichen
560  -          sta icr
570  -          lda #((sysirq) ;irq-vektor auf normalwert
580  -          sta irqvec
590  -          lda #)(sysirq)
600  -          sta irqvec+1
610  -          cli           ;irqs freigeben
```

Listing 8. Der dokumentierte Quellcode des

Verändern der Liste während des Programm-Ablaufs lassen sich vielfältige Effekte erzielen, wie Blinken, Farbscrolling, Bewegungseffekte oder ähnliches. Ich habe ein kleines Demoprogramm (Listing 9) geschrieben, das die Möglichkeiten der Routine demonstrieren soll. Einige der beschriebenen Effekte sind dort enthalten, und mit etwas Geschick läßt sich sicher noch einiges aus dem Programm herauskitzeln. Für die Assembler-Freaks gibt es noch den dokumentierten Quellcode (Listing 8). Zum Schluß noch einige Dinge, die Sie beachten sollten: Die Liste kann im Prinzip beliebig angelegt werden. Die Farbzonen dürfen sich jedoch nicht überschneiden und müssen in der Reihenfolge im Speicher stehen, wie sie auch auf dem Bildschirm erscheinen sollen. Also nicht die unterste Zone zuerst in der Liste. Die Zahlen in der Liste werden vom Programm nicht auf Richtigkeit überprüft, das muß von Ihrem Steuerprogramm erledigt werden. Außerdem darf die Länge der Liste 256 Zahlen nicht überschreiten. Da der VIC in jeder achten Rasterzeile wegen des Zeichenaufbaus die Rasterinterrupts verzögert, sollte ein Farbwechsel in Rasterzeilen, die der Formel z = 51 + 8 * x entsprechen, vermieden werden.

(K. Kähler/ef)

```
620 -          rts            ;ruecksprung ins basic
630 -;
640 -;***   hauptroutine   ***
650 -;
660 -irq       lda #$01        ;irq-request loeschen
670 -          sta irqreq
700 -;
710 -          jsr readtab    ;zonenfarbe 1 auslesen
720 -;
730 -;   hauptschleife
735 -;
740 -nxtzone   tay            ;nach y retten
750 -          ldx tabpnt     ;tabellenwert lesen
751 -          inc tabpnt     ;(aus zeitmangel
752 -          lda table,x    ;nicht als up)
760 -          clc            ;carry loeschen
770 -          adc rline      ;a:=akt. zeile + bereichsbreite
780 -          ldx rline      ;1 zeilenwechsel abwarten
782 -wait      cpx rline
784 -          beq wait
790 -          sty back       ;bildschirmfarbe wechseln
800 -wtend     cmp rline      ;warten, bis endzeile (in a) erreicht wird
810 -          bne wtend
820 -;
830 -;   naechster bereich/ende ?
840 -;
850 -          jsr readtab    ;naechsten wert holen
860 -          bpl nxtzone    ;wenn positiv,=) nxtzone
880 -;
885 -;   zonenende/listenende
890 -;
900 -setend    ldx rline      ;sonst 1 zeilenwechsel abwarten
902 -wait2     cpx rline
904 -          beq wait2
910 -strback   lda #$aa        ;bildschirmfarbe zurueckschreiben ($aa=dummy,
920 -          sta back       ;speicher fuer hintergrundfarbe)
925 -;
930 -          jsr readtab    ;naechsten zonenbeginn auslesen
940 -          beq again      ;wenn ende-kennung, =) again
945 -;
950 -          sta rline      ;sonst ausloesende zeile:=a
960 -          jmp irqout     ;irq beenden
970 -;
980 -again     sta tabpnt     ;tabellenzeiger:=0
990 -          jsr readtab    ;ersten zonenbeginn auslesen
1000 -         sta rline      ;ausloesende zeile:=a
1010 -         jmp sysirq     ;system-irq ausfuehren
1020 -;
1030 -;   up tabellenwert lesen
1040 -;
1050 -readtab   ldx tabpnt     ;tabpnt lesen
1060 -         inc tabpnt     ;tabellenzeiger erhoehen
1070 -         lda table,x    ;a:=tabellenstart+tabpnt
1080 -         rts            ;ruecksprung
1160 -;
1170 -;
1180 -;***   zonentabelle ab hier   ***
1190 -;
1200 -table     .by 99,2,3,6,5,2,3,128,0
```

»Screenmanagers«

```
1 IF A=0 THEN A=1:LOAD"SCRNMANAGER.OBJ",8,
   1                                      <008>
5 REM ****************************        <252>
10 REM *** SCREENMANAGER - DEMO ***       <022>
15 REM *** WRITTEN 13.11.87 BY   ***      <140>
16 REM *** K.KAEHLER             ***      <038>
17 REM *** SKALITZER STR.134A    ***      <255>
18 REM *** 1000 BERLIN 36        ***      <016>
20 REM ****************************       <011>
25 :                                      <001>
30 REM --- VARIABLEN ---                  <136>
35 :                                      <011>
40 TA=49300: REM TABELLENSTART            <165>
50 AN=49152: REM ANSCHALTEN               <073>
60 AUS=49193:REM AUSSCHALTEN              <071>
65 :                                      <041>
66 PO=170:FG=1:H=1                        <161>
70 FOR I=0 TO 8:READ CO(I):NEXT           <112>
71 :                                      <047>
72 REM --- ZONENDATEN EINLESEN ---        <207>
73 :                                      <049>
74 XX=TA                                  <114>
75 READ Q:IF Q>-1 THEN POKE XX,Q:XX=XX+1:G
   OTO 75                                 <097>
76 :                                      <052>
80 REM --- SCREEN AUFBAUEN ---            <234>
90 :                                      <066>
100 POKE 53280,0:POKE 49266,0:PRINT"{CLR,2
    DOWN,BLACK}"                          <109>
110 PRINT TAB(10)"** SCREENMANAGER **"    <120>
115 PRINT:PRINT"00000000000000000000000000
    00000000000000";                      <147>
120 PRINT"{RVSON,6SPACE}THE ULTIMATE SCREE
    N-UTILITY!{6SPACE,RVOFF}";            <225>
125 PRINT"UUUUUUUUUUUUUUUUUUUUUUUUUUUUUU
    UUUUUUUUUU"                           <002>
130 PRINT:PRINT TAB(12)"PROGRAM AND DEMO" <173>
140 PRINT:PRINT TAB(6)"WRITTEN IN 1987 BY
    K.KAEHLER"                            <056>
145 PRINT:PRINT:PRINT"{RVSON,40SPACE}";   <097>
150 PRINT"{14SPACE}C64 DEFEATED{14SPACE}"; <112>
160 PRINT"{13SPACE}AMIGA-COPPER!!{13SPACE}
    ";                                    <056>
165 PRINT"{40SPACE}";                     <165>
170 PRINT:PRINT"{RED,RVSON,10SPACE}>
    > SCREENMANAGER <<{11SPACE}"          <158>
240 :                                     <216>
250 REM --- EINSCHALTEN ---               <105>
260 :                                     <238>
270 SYS AN                                <120>
280 :                                     <002>
300 REM --- ANIMATION ---                 <122>
310 :                                     <032>
320 REM BLINKENDE BALKEN (#2 & #6)        <100>
325 :                                     <047>
330 FL=FL+1:IF FL>8 THEN FL=0             <043>
340 POKE 49311,CO(FL)                     <153>
350 POKE 49357,CO(FL)                     <167>
360 :                                     <082>
500 REM BEWEGLICHER BALKEN (#5)           <163>
510 :                                     <232>
520 PO=PO+FG                              <248>
530 IF PO<170 OR PO>188 THEN FG=-FG       <006>
540 POKE 49342,PO                         <071>
550 :                                     <018>
560 REM WANDERNDER FARBBALKEN(#4)         <170>
570 :                                     <038>
575 POKE 49325+H*2,6                      <102>
580 H=H+1:IF H>7 THEN H=1:RC=RND(0)*16    <103>
590 POKE 49325+H*2,RC                     <201>
799 GOTO 330                              <029>
800 :                                     <119>
801 REM --- FARBDATEN ---                 <004>
802 :                                     <016>
805 DATA 0,11,12,15,1,15,12,11,0          <248>
1000 :                                    <214>
1001 REM --- ZONENDATAS ---               <079>
1002 :                                    <216>
1030 DATA 48,11,4,12,3,15,3,1,3,128       <135>
1031 :                                    <247>
1032 DATA 70,0,12,128                     <121>
1033 :                                    <249>
1034 DATA 95,6,1,14,2,1,1,14,2,6,3,128    <080>
1035 :                                    <251>
```

Listing 9. Demo der Fähigkeiten des »Screenmanagers«

```
1036 DATA 122,6,5,6,6,6,5,6,5,6,5,6,5,6,5,
     128                                    <226>
1037 :                                      <253>
1038 DATA 177,11,2,12,2,1,1,12,2,11,2,128   <077>
1039 :                                      <255>
1040 DATA 214,6,3,0,7,6,3,128              <252>
1041 :                                      <001>
1050 DATA 235,1,3,15,3,12,3,11,3,128        <077>
2000 DATA 0,-1                              <032>
```

Listing 9. (Schluß)

```
Name : scrnmanager.obj    c000 c094
-----------------------------------
c000 : a9 00 85 fb 78 ad 11 d0  65
c008 : 29 7f 8d 11 d0 a9 7f 8d  ea
c010 : 0d dc a9 01 8d 1a d0 a9  56
c018 : 40 8d 14 03 a9 c0 8d 15  85
c020 : 03 20 8c c0 8d 12 d0 58  cc
c028 : 60 78 a9 00 8d 1a d0 a9  6f
c030 : 81 8d 0d dc a9 31 8d 14  d9
c038 : 03 a9 ea 8d 15 03 58 60  08
c040 : a9 01 8d 19 d0 20 8c c0  b2
c048 : a8 a6 fb e6 fb bd 94 c0  a0
c050 : 18 6d 12 d0 ae 12 d0 ec  56
c058 : 12 d0 f0 fb 8c 21 d0 cd  3f
c060 : 12 d0 d0 fb 20 8c c0 10  17
c068 : df ae 12 d0 ec 12 d0 f0  c1
c070 : fb a9 aa 8d 21 d0 20 8c  ce
c078 : c0 f0 06 8d 12 d0 4c 81  bf
c080 : ea 85 fb 20 8c c0 8d 12  59
c088 : d0 4c 31 ea a6 fb e6 fb  06
c090 : bd 94 c0 60 ee 41 03 a9  2c
```

Listing 10. »Screenmanager« verwaltet Rasterzeilen

7. Zeppo – ein Unterprogramm für Druckerfans

Das Programm »Zeppo« (Listing 11) gestattet, einen Text auf dem Drucker auszugeben – und zwar mit dem im Augenblick auf dem Bildschirm sichtbaren Zeichensatz. Dazu stehen vier Druckdichten zur Verfügung. Zeppo wurde für Epson-kompatible Drucker geschrieben. Es sind folgende Schritte erforderlich:

- Nach dem Abtippen mit dem MSE ist Zeppo absolut zu laden (LOAD "ZEPPO",8,1). Ein anschließendes NEW <RETURN> sorgt dafür, daß die Basic-Zeiger zurückgesetzt werden.
- Ist der gewünschte Zeichensatz aktiviert, muß zum Drucken folgende Syntax eingehalten werden:
  `SYS 50600,x,"string"<RETURN>`

Dabei entspricht »x« der Druckdichte. Es sind Zahlen von eins bis vier möglich. Statt »string« lassen sich auch Stringvariablen (z. B. a$) oder auch normale Variablen einsetzen. Feldvariablen sind nicht erlaubt. Auch sollte der String nicht länger als 255 Zeichen sein, da sonst ein Absturz zu befürchten ist. Außerdem ist dabei zu beachten, daß wegen der Druckerauflösung nur die ersten 60 Zeichen auf dem Papier erscheinen. Steuerzeichen wie CRSR-aufwärts-abwärts, HOME oder CLR sind nicht zugelassen. Wegen der Kürze des Programms wurden keinerlei Fehlerabfragen integriert. Daher stürzt Zeppo bei einer Fehlbedienung sang- und klanglos ab.

Diejenigen, die das Programm an einen anderen Drucker anpassen oder in ihren eigenen Programmen verwenden wollen, finden den entsprechenden Quellcode in Listing 12 (Format: Hypra-Ass). (Oliver Möller/ef)

```
Name : zeppo                c5a8 c6df
-----------------------------------
c5a8 : 20 fd ae 20 9e b7 86 6b  0f
c5b0 : e0 05 b0 04 e0 00 d0 01  13
c5b8 : 60 ad 88 02 8d 5c c6 20  68
c5c0 : fd ae a9 c7 8d 88 02 a2  e2
c5c8 : 00 a9 20 9d 00 c7 ca d0  64
c5d0 : fa 20 66 e5 20 a0 aa ad  3e
c5d8 : 1a d0 8d 61 c6 a9 f0 8d  82
c5e0 : 1a d0 78 a9 0c a2 04 a0  dd
c5e8 : 00 20 ba ff a9 00 20 bd  3d
c5f0 : ff 20 c0 ff a2 0c 20 c9  ce
c5f8 : ff a9 1b 20 d2 ff a6 6b  35
c600 : bd ce c6 20 d2 ff a6 6b  79
c608 : bd d2 c6 20 d2 ff a6 6b  83
c610 : bd d6 c6 20 d2 ff a9 3c  3a
c618 : 85 61 bd da c6 85 6b ad  ba
c620 : 00 dd 29 03 49 03 18 6a  9b
c628 : 6a 6a 85 69 a2 d0 ad 18  ed
c630 : d0 29 0e c9 04 f0 0b a2  8b
c638 : d8 c9 06 f0 05 0a 0a 05  67
c640 : 69 aa 8e 7f c6 a9 00 8d  67
c648 : 68 c6 a5 61 d0 19 a9 0d  3f
c650 : 20 d2 ff 20 cc ff a9 0c  69
c658 : 20 c3 ff a9 ff 8d 88 02  21
c660 : a9 ff 8d 1a d0 58 60 ad  5c
c668 : 00 c7 18 a2 00 2a a8 8a  af
c670 : 2a aa 98 2a a8 8a 2a aa  38
c678 : 98 2a a8 8a 2a 18 69 ff  aa
c680 : 85 6e 84 6d a2 00 a0 00  b8
c688 : 78 a9 31 85 01 84 6c b1  1b
c690 : 6d 3d c7 c6 f0 03 b9 c7  04
c698 : c6 05 6c 85 6c c8 c0 08  cd
c6a0 : d0 ed a4 6b 84 6a 86 69  86
c6a8 : a9 37 85 01 58 a2 01 a5  58
c6b0 : 6c 20 d2 ff c6 6a d0 f5  d0
c6b8 : a6 69 e8 e0 08 d0 c7 c6  1d
c6c0 : 61 ee 68 c6 4c 4a c6 80  be
c6c8 : 40 20 10 08 04 02 01 4b  08
c6d0 : 59 4c 5a e0 c0 c0 80 01  18
c6d8 : 03 03 07 01 02 02 04 00  7f
```

Listing 11. »Zeppo« – ein Programm für Druckerfans

```
6    -.EQ SCREEN=$C700
8    -.EQ SID=54272
12   -.EQ COUNT2=COUNT1+1
16   -.EQ GETBYT=$B79E
18   -.EQ CHKCOM=$AEFD
20   -.EQ CHRGOT=$0079
22   -.EQ CHRGET=$0073
24   -.EQ ERROR=$A437
26   -.EQ PRINT=$AB1E
28   -.EQ OPEN=$FFC0
30   -.EQ SETFLS=$FFBA
32   -.EQ CHKOUT=$FFC9
34   -.EQ CLRCH=$FFCC
36   -.EQ CLOSE=$FFC3
38   -.EQ PINS=$FFD2
40   -.EQ DRUCK=$AAA0
42   -.EQ HOME=$E566
44   -.EQ QUEST=$AB45
46   -.EQ SETNAM=$FFBD
50   -;
52   -.EQ ZW1=$69 ;
54   -.EQ ZW2=$6A ;
56   -.EQ NUME=$6B;
58   -.EQ SIGN=$6C; IM ARG
60   -.EQ Z1=$6D  ;
62   -;ZERO-ADRESSE
64   -;
66   -.EQ COUNT1=$61;ZAEHLER
70   -;---
90   -.OB"ZEPPO,P,W"
99   -.BA $C5A8;50600
100  -          JSR CHKCOM     ;FOLGT KOMMA?
110  -          JSR GETBYT     ;DRUCKDICHTE HOLEN
120  -          STX NUME
130  -;
140  -          CPX #5         ;SICHERHEITS-
150  -          BCS ERR
160  -          CPX #0         ;ABFRAGE
170  -          BNE OK
180  -ERR       RTS
190  -;
200  -OK        LDA 648        ;BILDSCHIRM
210  -          STA SCMERK+1   ;MERKEN
220  -;
230  -          JSR CHKCOM     ;2.KOMMA
240  -          LDA #>(SCREEN)
250  -          STA 648        ;ERSATZ-BILDSCHIRM
260  -          LDX #0
270  -          LDA #32
280  -RMKT      STA SCREEN,X   ;Z.T.LOESCHEN
290  -          DEX
300  -          BNE RMKT
310  -;
320  -          JSR HOME
330  -          JSR DRUCK      ;"PRINT"-BEFEHL AUSFUEHREN
340  -;
350  -;
360  -          LDA $D01A
365  -          STA RIRM+1
368  -          LDA #$F0
369  -          STA $D01A      ;VIC-INTERRUPT SPERREN
370  -;
375  -          SEI            ;KEIN INTERRUPT
380  -          LDA #12        ;LOG. FILENUMMER
390  -          LDX #4         ;GERAET=DRUCKER
400  -          LDY #0         ;SEK.AD
410  -          JSR SETFLS     ;SETZEN
420  -          LDA #0         ;KEIN'FILENAME'
430  -          JSR SETNAM
440  -;
450  -          JSR OPEN
460  -          LDX #12
470  -          JSR CHKOUT     ;AUSGABE AUF DATEI
480  -;
490  -;
500  -          LDA #27        ;ESCAPE
```

Listing 12. Quellcode »Zeppo« (Format: Hypra-Ass) ▶

```
510  -           JSR PINS
520  -           LDX NUME        ;\
530  -           LDA DAT1-1,X    ; |
540  -           JSR PINS        ; CODE
550  -           LDX NUME        ; |  FUER
560  -           LDA DAT2-1,X    ; DRUCKER-
570  -           JSR PINS        ; |
580  -           LDX NUME        ; |
590  -           LDA DAT3-1,X    ; | (EPSON)
600  -           JSR PINS        ;/
610  -;
620  -MERKER     LDA #60         ;60 ZEICHEN
630  -           STA COUNT1
640  -;
650  -           LDA DAT4-1,X
660  -           STA NUME
670  -           LDA $DD00
680  -           AND #3
690  -           EOR #3
700  -           CLC
710  -           ROR
720  -           ROR
730  -           ROR
740  -           STA ZW1
750  -;
760  -           LDX #$D0        ;ZEICHEN-ROM
770  -           LDA $D018
780  -           AND #14
790  -           CMP #4          ;NORMAL?
800  -           BEQ NOR
810  -;
820  -           LDX #$D8        ;ZEICHEN-ROM (KLEINSCHRIFT
830  -           CMP #6          ;NORMALE KLEINSCHRIFT ?
840  -           BEQ NOR
850  -;
860  -           ASL
870  -           ASL             ;2  BIT NACH LINKS
880  -           ORA ZW1
890  -           TAX
900  -NOR        STX ZEIC+1
910  -;---------------------
920  -START      LDA #0
930  -           STA AUSGABE+1
940  -;
950  -KNAK       LDA COUNT1
960  -           BNE AUSGABE
970  -;
980  -           LDA #13         ;ZEILENVORSCHUB
990  -           JSR PINS
1000 -           JSR CLRCH       ;KEINE DRUCKERAUSGABE MEHR
1010 -           LDA #12         ;FL-NUMME
1020 -           JSR CLOSE
1030 -SCMERK     LDA #$FF        ;DUMMY
1040 -           STA 648
1042 -RIRM       LDA #$FF        ;DUMMY
1046 -           STA $D01A       ;VIC-INTERRUPT ERMOEGLICHE
1050 -           CLI
1060 -           RTS             ;R UECKSPRUNG
1070 -;
1080 -AUSGABE    LDA SCREEN
1090 -           CLC
1100 -           LDX #0
1110 -           ROL
1120 -           TAY
1130 -           TXA
1140 -           ROL
1150 -           TAX
1160 -           TYA
1170 -           ROL
1180 -           TAY
1190 -           TXA
1200 -           ROL
1210 -           TAX
1220 -           TYA
1230 -           ROL
1240 -           TAY
1250 -           TXA
1260 -           ROL                  ;* 8
1270 -;
1280 -           CLC
1290 -ZEIC       ADC #$FF             ;DUMMY
1300 -           STA Z1+1
1310 -           STY Z1
1320 -;
1330 -;
1340 -           LDX #0
1350 -;
1360 -MARK3      LDY #0
1370 -           SEI
1380 -           LDA #$31
1390 -           STA 1                ; RAM-KONF.
1400 -           STY SIGN
1410 -MARK       LDA (Z1),Y           ;\
1420 -           AND BITDAT,X         ; |
1430 -           BEQ NORP             ; |
1440 -           LDA BITDAT,Y         ; |  BYTE
1450 -NORP       ORA SIGN             ; |-
1460 -           STA SIGN             ; |  KIPPEN
1470 -           INY                  ; |
1480 -           CPY #8               ; |
1490 -           BNE MARK             ;/
1500 -           LDY NUME
1510 -           STY ZW2
1520 -           STX ZW1
1530 -;
1540 -           LDA #$37
1550 -           STA 1
1560 -           CLI
1570 -;
1580 -MARK2      LDX #1
1590 -           LDA SIGN
1600 -           JSR PINS             ;AUSDRUCK
1610 -           DEC ZW2              ;DICHTE
1620 -           BNE MARK2
1630 -           LDX ZW1
1640 -           INX                  ;NAECHSTE PIXEL-REIHE
1650 -           CPX #8
1660 -           BNE MARK3
1670 -;
1680 -;
1690 -           DEC COUNT1
1700 -           INC AUSGABE+1
1710 -           JMP KNAK             ;RUECKSPRUNG
10000-BITDAT     .BY 128,64,32,16,8,4,2,1
10010-DAT1       .BY $4B,$59,$4C,$5A
10020-DAT2       .BY $E0,$C0,$C0,$80
10030-DAT3       .BY 1,3,3,7
10040-DAT4       .BY 1,2,2,4
20000-.EN
40000-TEXT
50000-;
50001-;
50002-;
50003-; EIN PROGRAMM VON OLIVER MOELLER
50004-;
50005-;
50006-;
```

Listing 12. (Schluß)

8. 112 Sprites

Diese Routine macht es möglich, 112 Sprites in Form einer 8 * 14-Matrix auf den Bildschirm des C 64 zu bringen. Auf die Bewegung der Sprites wurde verzichtet, da nur das Prinzip der Sprite-Darstellung verständlich werden soll. Vom C 64 ist bekannt, daß acht Sprites ohne Probleme darstellbar sind. Durch geschickte Umschaltung der Sprites durch den Rasterzeileninterrupt lassen sich bis zu 14 Zeilen mit jeweils acht Sprites einblenden. Bevor man 112 Sprites sehen kann, ist Listing 13 einzugeben. Der Aufruf der Routine erfolgt über SYS 8263. Listing 14 zeigt den dokumentierten Quellcode.

(M. Herrmann/ef)

```
Name : 112 sprites          2000 212d
-----------------------------------
2000 : 00 00 00 7f ff fe 7f ff  e6
2008 : fe 7f ff fe 7f ff fe 7f  98
2010 : ff fe 7f ff fe 7f ff fe  58
2018 : 7f ff fe 7f ff fe 7f ff  3c
2020 : fe 7f ff fe 7f ff fe 7f  b0
2028 : ff fe 7f ff fe 7f ff fe  70
2030 : 7f ff fe 7f ff fe 7f ff  54
2038 : fe 7f ff fe 00 00 00 0b  ec
2040 : 0b 0c 0f 0f 0c 0b 0b 78  2d
2048 : a9 7f 8d 0d dc a9 f1 8d  b4
2050 : 1a d0 a9 00 8d 12 d0 8d  04
2058 : 20 d0 8d 21 d0 a9 1b 8d  4a
2060 : 11 d0 a9 20 8d 15 03 a9  29
2068 : 80 8d 10 d0 a9 ff 8d 15  c8
2070 : d0 20 44 e5 20 66 e5 a2  30
2078 : 00 a9 80 9d f8 07 bd 3f  5e
2080 : 20 9d 27 d0 e8 e0 08 d0  aa
2088 : f0 a9 01 8d 86 02 a9 97  8d
2090 : 8d 14 03 58 60 ea ea ce  9a
2098 : 19 d0 a9 f4 8d 12 d0 a9  22
20a0 : 1b 8d 11 d0 a2 20 ca d0  d8
20a8 : fd a9 13 8d 11 d0 a2 00  12
20b0 : bd fd 20 9d 00 d0 e8 e0  14
20b8 : 10 d0 f5 a2 a0 ca d0 fd  a2
20c0 : a2 00 bd 0d 21 9d 00 d0  14
20c8 : e8 e0 10 d0 f5 a2 ff ca  48
20d0 : d0 fd a2 00 bd 1d 21 9d  cc
20d8 : 00 d0 e8 e0 10 d0 f5 a0  37
20e0 : 0b a2 8b ca d0 fd a2 01  02
20e8 : bd 00 d0 18 69 15 9d 00  92
20f0 : d0 e8 e8 e0 11 d0 f1 88  fb
20f8 : d0 e7 4c 31 ea 58 01 70  4b
2100 : 01 88 01 a0 01 b8 01 d0  15
```

**Listing 13.
So bringt man 112 Sprites auf den Bildschirm**

Listing 13. (Schluß)

```
2108 : 01 e8 01 00 01 58 16 70  ca
2110 : 16 88 16 a0 16 b8 16 d0  25
2118 : 16 e8 16 00 16 58 05 70  41
2120 : 05 88 05 a0 05 b8 05 d0  8b
2128 : 05 e8 05 00 05 29 7f 91  9d
```

```
-       .ba$2000

-; Spritedaten

-12000 .by $00,$00,$00,$7f,$ff,$fe,$7f,$ff
-      .by $fe,$7f,$ff,$fe,$7f,$ff,$fe,$7f
-      .by $ff,$fe,$7f,$ff,$fe,$7f,$ff,$fe
-      .by $7f,$ff,$fe,$7f,$ff,$fe,$7f,$ff
-      .by $fe,$7f,$ff,$fe,$7f,$ff,$fe,$7f
-      .by $ff,$fe,$7f,$ff,$fe,$7f,$ff,$fe
-      .by $7f,$ff,$fe,$7f,$ff,$fe,$7f,$ff
-      .by $fe,$7f,$ff,$fe,$00,$00,$00

-; Spritefarben

-1203f .by $0b,$0b,$0c,$0f,$0f,$0c,$0b,$0b

-; Routine zum Einbinden

-12047 sei            ; Interrupt sperren
-      lda #$7f
-      sta $dc0d      ; CIA-Interrupt löschen
-      lda #$f1       ; Rasterzeileninterrupt
-      sta $d01a      ; setzen
-      lda #$00
-      sta $d012      ; Rasterzeile setzen
-      sta $d020      ; Rahmen      : Schwarz
-      sta $d021      ; Hintergrund: Schwarz
-      lda #$1b
-      sta $d011      ; Textbildschirm
-      lda #$20       ; IRQ-Vektor (high)
-      sta $0315      ; setzen
-      lda #$80       ; Position der Sprites
-      sta $d010      ; (Bits 8)
-      lda #$ff
-      sta $d015      ; Sprites einschalten
-      jsr $e544      ; Bildschirm löschen
-      jsr $e566      ; Cursor nach oben
-      ldx #$00
-12079 lda #$80       ; Spritezeiger
-      sta $07f8,x;   setzen
-      lda 1203f,x;   Farben der
-      sta $d027,x;   Sprites setzen
-      inx
-      cpx #$08       ; für acht Sprites
-      bne 12079
-      lda #$01       ; Schriftfarbe: Weiß
-      sta $0286
-      lda #$97       ; IRQ-Vektor (low)
-      sta $0314      ; setzen
-      cli            ; Interrupt zulassen
-      rts
-      nop            ; Leerbytes für JMP
-      nop

-; Interruptroutine

-      dec $d019      ; IRQ-Flag löschen
-      lda #$f4       ; Neue Rasterzeile
-      sta $d012      ; setzen
-      lda #$1b       ; Bildschirm auf 24
-      sta $d011      ; Zeilen verkleinern
-      ldx #$20
-120a6 dex            ; Verzögerung
-      bne 120a6
-      lda #$13       ; Bildschirm auf 25
-      sta $d011      ; Zeilen vergrößern
-      ldx #$00
-120b0 lda 120fd,x;   Spritepositionen
-      sta $d000,x;   setzen
-      inx
-      cpx #$10       ; 16 Byte
-      bne 120b0
-      ldx #$a0
-120bd dex            ; Verzögerung
-      bne 120bd
-      ldx #$00
-120c2 lda 1210d,x;   Spritepositionen
-      sta $d000,x;   setzen
-      inx
-      cpx #$10       ; 16 Byte
-      bne 120c2
-      ldx #$ff
-120cf dex            ; Verzögerung
-      bne 120cf
-      ldx #$00
-120d4 lda 1211d,x;   Spritepositionen
-      sta $d000,x;   setzen
-      inx
-      cpx #$10
-      bne 120d4
-      ldy #$0b       ; 11 Reihen Sprites
-120e1 ldx #$8b
-120e3 dex            ; Verzögerung
-      bne 120e3
-      ldx #$01       ; Zähler setzen
-120e8 lda $d000,x;   Spriteposition lesen
-      clc
-      adc #$15       ; 21 addieren
-      sta $d000,x;   und neu setzen
-      inx
-      inx
-      cpx #$11       ; 8 Sprites
-      bne 120e8
-      dey            ; 11 Reihen Sprites
-      bne 120e1
-      jmp $ea31      ; Normale IRQ-Routine

-; Spritepositionen

-120fd .by $58,$01,$70,$01,$88,$01,$a0,$01
-      .by $b8,$01,$d0,$01,$e8,$01,$00,$01
-1210d .by $58,$16,$70,$16,$88,$16,$a0,$16
-      .by $b8,$16,$d0,$16,$e8,$16,$00,$16
-1211d .by $58,$05,$70,$05,$88,$05,$a0,$05
-      .by $b8,$05,$d0,$05,$e8,$05,$00,$05
```

Listing 14. Der Quelltext zur Spriteroutine

9. Super-Rasterzeilen-Interrupt

Es gibt mittlerweile viele Methoden, den Bildschirm durch ausgefeilte Interrupttechnik aufzuteilen oder ihn von seinem Rahmen zu befreien. Häufig ist es nötig, einen Rasterzeilen-Interrupt an einer genau definierten X-Position erfolgen zu lassen, weil sich nur so zum Beispiel der seitliche Rahmen entfernen läßt. Bislang ließ sich nur mit entsprechend vielen NOP-Befehlen das Zeitverhalten so austüfteln, daß kein Flimmern auftritt. An ein parallel arbeitendes Programm war gar nicht zu denken, da durch unterschiedliche Befehlslängen der Interrupt bis zu sieben Mikrosekunden zu spät kommen kann, was das Timing komplett durcheinanderwirft. Das Listing 15 zeigt jedoch eine Routine, die am oberen Rand des Bildschirms einen kurzen Farbstreifen darstellt, der sich durch (fast) nichts von seiner Lage abbringen läßt. Darüber befindet sich ein Streifen, der nach der »herkömmlichen« Methode gebildet wird und bei Tastendrücken dementsprechend stark hin und her springt. Nun jedoch zur Beschreibung des Programms:

Nach dem Aufruf des Programms mit

```
SYS 49152
```

wird der Rasterzeilen-Interrupt initialisiert und ins Hauptprogramm zurückgesprungen. Die Interrupt-Routine beginnt ab $C020 (siehe Quelltext, Listing 16).

Zunächst wird das IRQ-Register des Videochips gelöscht. Dann erfolgt durch entsprechend viele NOP-Befeh-

```
Name : super raster irq  c000 c0b0
----------------------------------
c000 : 78 a9 20 8d 14 03 a9 c0  88
c008 : 8d 15 03 a9 01 8d 1a d0  9c
c010 : 8d 0d dc a9 28 8d 12 d0  69
c018 : a9 1b 8d 11 d0 58 60 00  26
c020 : ad 19 d0 8d 19 d0 ea ea  d9
c028 : ea ea ea ea ea ea ea a2  96
c030 : ff a0 00 8e 00 dc 8c 02  6e
c038 : dc 8e 03 dc 8e 01 dc 8c  35
c040 : 01 dc 8e 01 dc 8c 20 d0  c7
c048 : a9 0e 8d 20 d0 ad 13 d0  c8
c050 : 8e 02 dc 8c 03 dc 8e 01  fb
c058 : dc a2 7f 8e 00 dc 4a 4a  dc
c060 : 4a 8d 68 c0 90 00 b8 50  30
c068 : 12 ea ea ea ea ea ea ea  8f
c070 : ea ea ea ea ea ea ea ea  6f
c078 : ea ea ea ea ea ea ea ea  77
c080 : ea ea ea ea ea ea ea ea  7f
c088 : ea ea ea ea ea ea ea ea  87
c090 : ea ea ea ea ea ea ea ea  8f
c098 : ea ea a2 09 ca d0 fd ea  c2
c0a0 : ea ea a9 01 8d 20 d0 a9  fa
c0a8 : 0e 8d 20 d0 4c 31 ea aa  ee
```

Listing 15. Und es geht doch flackerfrei – der horizontale Rasterzeilen-Interrupt

le eine Verzögerung, die dafür sorgt, daß der Farbstreifen ungefähr in der Mitte des Bildschirms steht. Von Adresse $C045 bis $C04C wird dieser Farbstreifen erzeugt.

Doch wozu nun die vielen STX und STY in die Register ab $DC00? Zunächst wird hier Port A der CIA 1 auf Eingabe, dann Port B auf Ausgabe geschaltet. Nun werden in Port B alle Bits gesetzt, gelöscht und wieder gesetzt. »Was soll das denn jetzt schon wieder?« werden Sie fragen. Kenner der C 64-Hardware wissen, daß am Joystick-Port 1 ein Eingang für einen Lightpen vorhanden ist. Dieser ist mit dem Eingang für den Feuerknopf identisch. Wird nun der Eingang für den Feuerknopf als Ausgang programmiert, so lassen sich über den CIA Impulse auf den Lightpen-Eingang des VIC geben. Der Videochip legt bei einem Interrupt, der ja hier durch die CIA 1 erzeugt wird, die aktuelle Position des Elektronenstrahls in Register $D013 ab. Jetzt wird der Urzustand des Tastatur-Ports wiederhergestellt.

Im Akku befindet sich nun die X-Position, an der in Port B alle Bits auf 0 gingen. Die Genauigkeit beträgt zwei Bildpunkte. Die kleinste Zeiteinheit, mit der der C 64 einen Pro-

```
,c000  78        sei        ; interrupt sperren
,c001  a9 20     lda #20    ; interruptvektor
,c003  8d 14 03  sta 0314   ; auf die eigene
,c006  a9 c0     lda #c0    ; routine richten
,c008  8d 15 03  sta 0315
,c00b  a9 01     lda #01
,c00d  8d 1a d0  sta d01a
,c010  8d 0d dc  sta dc0d
,c013  a9 28     lda #28
,c015  8d 12 d0  sta d012
,c018  a9 1b     lda #1b
,c01a  8d 11 d0  sta d011
,c01d  58        cli        ; interrupt freigeben
,c01e  60        rts
-----------------------------------------------
,c01f  00        brk        ; leerbyte
-----------------------------------------------
,c020  ad 19 d0  lda d019   ; interrupt-flag
,c023  8d 19 d0  sta d019   ; löschen
,c026  ea        nop
,c027  ea        nop
,c028  ea        nop
,c029  ea        nop
,c02a  ea        nop
,c02b  ea        nop
,c02c  ea        nop
,c02d  ea        nop
,c02e  ea        nop
,c02f  a2 ff     ldx #ff
,c031  a0 00     ldy #00
,c033  8e 00 dc  stx dc00    ;lightpen-
,c036  8c 02 dc  sty dc02    ;interrupt-
,c039  8e 03 dc  stx dc03    ;anforderung
,c03c  8e 01 dc  stx dc01    ;position
,c03f  8c 01 dc  sty dc01    ;nun in
,c042  8e 01 dc  stx dc01    ;$D013
,c045  8c 20 d0  sty d020
,c048  a9 0e     lda #0e     ;herkömm-
,c04a  8d 20 d0  sta d020    ;licher
,c04d  ad 13 d0  lda d013    ;Balken
,c050  8e 02 dc  stx dc02
,c053  8c 03 dc  sty dc03
,c056  8e 01 dc  stx dc01
,c059  a2 7f     ldx #7f
,c05b  8e 00 dc  stx dc00    ;pos. ist
,c05e  4a        lsr         ;im akku
,c05f  4a        lsr
,c060  4a        lsr
,c061  8d 68 c0  sta c068
,c064  90 00     bcc c066    ;timing
,c066  b8        clv         ;ausrechnen
,c067  50 12     bvc c07b
,c069  ea        nop
,c06a  ea        nop
,c06b  ea        nop
,c06c  ea        nop
,c06d  ea        nop
,c06e  ea        nop
,c06f  ea        nop
,c070  ea        nop
,c071  ea        nop
,c072  ea        nop
,c073  ea        nop
,c074  ea        nop
,c075  ea        nop
,c076  ea        nop
,c077  ea        nop
,c078  ea        nop
,c079  ea        nop
,c07a  ea        nop
,c07b  ea        nop
,c07c  ea        nop
,c07d  ea        nop
,c07e  ea        nop
,c07f  ea        nop
,c080  ea        nop
,c081  ea        nop
,c082  ea        nop
,c083  ea        nop
,c084  ea        nop
,c085  ea        nop
,c086  ea        nop
,c087  ea        nop
,c088  ea        nop
,c089  ea        nop
,c08a  ea        nop
,c08b  ea        nop
,c08c  ea        nop
,c08d  ea        nop
,c08e  ea        nop
,c08f  ea        nop
,c090  ea        nop
,c091  ea        nop
,c092  ea        nop
,c093  ea        nop
,c094  ea        nop
,c095  ea        nop
,c096  ea        nop
,c097  ea        nop
,c098  ea        nop
,c099  ea        nop
,c09a  a2 09     ldx #09
,c09c  ca        dex         ;verzögerung
,c09d  d0 fd     bne c09c
,c09f  ea        nop
,c0a0  ea        nop
,c0a1  ea        nop
,c0a2  a9 01     lda #01     ;weißen balken
,c0a4  8d 20 d0  sta d020    ;erzeugen
,c0a7  a9 0e     lda #0e     ;normale
,c0a9  8d 20 d0  sta d020    ;hintergrundfarbe
,c0ac  4c 31 ea  jmp ea31    ;normaler interrupt
```


Listing 16. Der Quelltext hilft beim Verständnis des erläuternden Textes

zeß steuern kann, beträgt ungefähr eine Mikrosekunde (ein Taktzyklus). In dieser Zeit »schreibt« der VIC jedoch acht Punkte auf den Bildschirm. Der Wert im Akku muß demnach durch vier geteilt werden, um den Abstand vom linken Bildschirmrand in Taktzyklen zu erhalten (Adressen $C05E und $C05F).

Die nächsten Programmschritte halten den Prozessor eine Zeitlang auf. Die Anzahl ergibt sich aus der Differenz der Zahl 64 und dem Inhalt des Akkus. So wird der Prozessor mit dem Rasterstrahl genau synchronisiert. Da es keine Befehle mit einem Taktzyklus Dauer gibt, muß wieder ein Trick her. Der Akku wird noch einmal durch zwei geteilt ($C060) und in Adresse $C068 geschrieben. Dort steht aber der relative Operant eines BVC-Befehls, der durch das CLV davor zu einem unbedingten Sprung wird. Dieser Branch-Befehl springt nun Akkuinhalt/2 Bytes weit. Da sein Sprungziel innerhalb von 32 NOPs liegt und ein NOP zwei Taktzyklen zur Ausführung benötigt, »paßt« das genau auf unsere Anforderungen. Beispiel: Stand im Akku die Zahl dezimal 10, so werden nun 5 NOPs übersprungen - das dauert genau unsere erforderlichen 10 Taktzyklen.

Doch was ist mit ungeraden Akkuinhalten? Geht nicht beim Teilen durch zwei das letzte Bit verloren, so daß eine Ungenauigkeit von einem Taktzyklus auftreten kann? Nicht ganz, denn das herausgeschobene Bit steht noch im Carry-Flag des Prozessor-Status-Registers - eine wichtige Eigenheit des LSR-Befehls. Bei gesetztem Carry-Flag soll unser Programm also einen Taktzyklus weniger warten als bei einem gelöschten. Die entsprechende Befehlsfolge steht ab Adresse $C064 und besteht nur aus einem BCC-Befehl, der auf das ihm folgende Clear Overflow (CLV) weist. Dieser unsinnig erscheinende Befehl hat die für unsere Zwecke nötige Eigenschaft: Springt der Prozessor nicht (Bedingung nicht erfüllt, Carry-Bit gesetzt), so braucht er zwei Taktzyklen, springt er (Bedingung erfüllt, Carry gelöscht), so benötigt er drei Taktzyklen. Das Verb »springen« mag hier irritieren, da der Prozessor in jedem Fall bei dem CLV weiterarbeitet - der geringe Unterschied ist zum exakten Timing jedoch unerläßlich.

Der Rest ist einfach. Nach den NOPs kommt eine Verzögerungsschleife, die den weißen Streifen positioniert ($C09A bis $C0A1), selbiger wird erzeugt ($C0A2 bis $C0AB) und in den System-Interrupt verzweigt - fertig!

Bei Betätigung einer beliebigen Taste flackert der obere Balken sofort, während der untere, nach unserem neuen Prinzip erzeugte, fest steht wie einbetoniert - mit einer Ausnahme: Ein angeschlossener Lichtgriffel, ein gedrückter Joystick-Feuerknopf oder die Betätigung der Space-Taste (alles eine Steuerleitung) bringen unsere Routine durcheinander, da ein Interrupt angefordert wird.

Dieser kleine Trick mit großer Wirkung eröffnet neue Perspektiven in der Raster-Interrupt-Programmierung. Wie wäre es mit neuen Mustern im oberen und unteren Rahmen (man denke auch an Speicherzelle $3FFF) oder mit einer interessanten Aufteilung des Bildschirms - links Grafik, rechts Text? Ein gelungenes Demo befindet sich auf der Programmservice-Diskette zu dieser Ausgabe. Ein weites Feld tut sich den Profis auf... (A. Beermann/ef)

10. Die 26ste Zeile

»Generator 26« reizt den Videochip des C64 wieder etwas weiter aus. Wie der Name schon fast sagt, erzeugt das Programm eine 26ste Zeile auf dem Bildschirm.

Damit die Theorie nicht allzu langweilig wird, tippen Sie bitte zunächst Listing 17 mit dem MSE (Seite 159) ab. Starten Sie es mit:

```
LOAD"GENERATOR 26",8,1
NEW
SYS 49152
```

Sie befinden sich nun im 26-Zeilen-Modus des C64. Der obere und der untere Rand müßten nun ein wenig flimmern, unter Umständen sind über diesen beiden Rändern schwarze, senkrechte Streifen zu sehen, doch dazu später. Es folgt zunächst eine Auflistung aller Befehle, die zum Bedienen des Programms notwendig und hilfreich sind. Die zusätzliche Zeile wird im folgenden Zeile 0 genannt.

SYS 49152 initialisiert das ganze Programm, Zeile 0 wird dabei gelöscht. Dies sollte immer als erstes nach dem Laden eingegeben werden.

SYS 49155 wirkt wie SYS 49152, außer daß Zeile 0 nicht gelöscht wird.

SYS 49161 wirkt wie SYS 49155, außer daß das Basic-ROM und das Kernel-ROM nicht in das darunter liegende RAM kopiert werden. Doch Vorsicht: Es entstehen dabei Einschränkungen bei der Benutzung von Zeile 0 (siehe »POKE 1,55«).

SYS 49179 schaltet den 26-Zeilen-Modus wieder aus. Dies ist zum Beispiel für das Laden von Diskette nötig. Versucht man im 26-Zeilen-Modus zu laden, führt dies unweigerlich zum Absturz.

SYS 49168 löscht Zeile 0.

SYS 49369, String schreibt den String (maximal 40 Zeichen) in Zeile 0 ab Spalte 0.

SYS 49358, Spalte, String schreibt den String in Zeile 0 ab der angegebenen Spalte.

POKE 1, 55 »nagelt« den Inhalt von Zeile 0 praktisch fest. Der Inhalt dieser Zeile kann dann weder durch Scrollen oder Löschen des Bildschirms geändert werden. Er kann nur noch durch die oben genannten SYS-Befehle manipuliert werden.

POKE 1, 53 hebt den unter »POKE 1, 55« genannten Zustand wieder auf. Vorsicht: Es muß sichergestellt sein, daß das RAM unter dem ROM nicht verändert wurde.

POKE 49504, Zahl blendet den oberen und unteren Rahmen aus, wenn Zahl=128 gilt. Liegt der Parameter zwischen 0 und 15, nehmen der obere und der untere Rahmen die Farbe von »Zahl« an (Farb-Codes im Bedienungshandbuch auf Seite 139).

Beachten Sie bitte, daß Programme im 26-Zeilen-Modus zirka 50 Prozent langsamer abgearbeitet werden. Des weiteren sollte man den neuen Modus nicht zu aktivieren versuchen, wenn er bereits aktiv ist. Ein Absturz des C64 ist sonst nicht in jedem Fall zu verhindern.

Eventuell erscheinen schwarze Streifen am oberen und unteren Bildschirmrand. Hauptschuldig an diesem Effekt ist die Speicherzelle $3FFF. Das »Warum« ist bereits in der Ausgabe 1/88 des 64'er-Magazins auf Seite 74 beantwortet worden. Abhilfe leistet jedenfalls ein »POKE 4*4096-1,0«, was aber wieder ein Basic-Programm überschreiben kann. Dann hilft nur eine lange REM-Zeile, die kurz vor der Adresse $4000 beginnt, so daß der POKE-Befehl allein den Kommentar zerstört. Ist Ihr Programm kürzer als 14 KByte, so sollte der Programmende-Zeiger (45/46) auf $4000 gerichtet werden, damit auch Variablen ungefährdet sind.

Wichtig zu wissen ist außerdem noch, daß das Programm bei der Bearbeitung von Strings die Länge nicht kontrolliert. Programmierer, die ihre Programme hinter Generator 26 (ab $C328) legen, sollten besonders darauf achten, daß die zu bearbeitenden Strings bei einem »SYS 49369, String«-Befehl nicht über 40 Zeichen lang sind. Bei einem »SYS 49358, X, (String)« darf der String nicht über 40 minus X Zeichen lang sein.

Zu beachten ist dabei noch die Tatsache, daß vom Pro-

gramm <RVS ON>- und <RVS OFF>-Codes in einem String nicht (!) als ein Zeichen zählt, die Codes aber ausführt. Folgender String ist also unbedenklich einsetzbar:

```
A$ = "0123456789<RVS ON>01234567890123456789<RVS
        OFF>0123456789"
```

Weiterhin behält <CRSR RIGHT> in einem String seine Funktion, ein Zeichen auf dem Bildschirm zu überspringen (zählt als ein Zeichen). <CRSR LEFT>, <CRSR UP> und <CRSR DOWN> werden nicht ausgeführt, wohl aber im Bildschirmcode angezeigt. Ein CHR$ (13) oder CHR$ (141) in einem String führt zum Abbruch des Ausdrucks in Zeile 0. Auf der Programmservice-Diskette zu dieser Ausgabe finden Sie zusätzlich noch ein kleines Demo zu Generator 26.

Grundlagen

Das Prinzip der 26sten Zeile beruht darauf, daß der Bildschirm des C64 um 7 Bit in vertikaler Richtung verschoben werden kann. Die Hauptroutine »hängt« im Raster-Interrupt. Um diese zusätzliche Zeile sichtbar zu machen, müssen der obere und untere Bildschirmrand ausgeblendet werden, es kann aber ein künstlicher Rand auf Wunsch erzeugt werden. Beim Aufruf mit einem SYS wird der Raster-Interrupt initialisiert, der erste wird bei Rasterzeile $F9 gesetzt. Im folgenden wird beschrieben, was bei den einzelnen Interrupts passiert:

Zeile $F9: Die Vorbereitungen zum Ausblenden des Randes werden getroffen und der nächste Raster-Interrupt auf $FF festgelegt.

Zeile $FF: Der obere und untere Bildschirmrahmen werden ausgeblendet und der Bildschirm um 7 Bit nach oben verschoben. Die Daten aus Zeile 1 ($0400 bis $0427) werden zwischengespeichert (von $C2D0 bis $C2F7), und die Daten, welche Zeile 0 repräsentieren, werden aus dem Bereich $C300 bis $C327 in Zeile 1 kopiert. Dann durchläuft das Programm eine Warteschleife, bis der Rasterstrahl oben am Bildschirm bei $30 angelangt ist (daraus resultiert der knapp 50prozentige Geschwindigkeitsverlust, da in dieser Zeit ein Basic-Programm nicht abgearbeitet werden kann).

Zeile $30: Diese Rasterzeile ist für den VIC wichtig: Hier holt er sich nämlich die Daten aus dem Bildschirmspeicher für die nächsten acht Rasterzeilen. Sobald er das getan hat, wird der Bildschirm um 7 Bit nach unten verschoben und die Daten aus dem Zwischenspeicher für Zeile 1 ($C2D0 bis $C2F7) wieder in den Speicher für Zeile 1 geschrieben, wo vorher der VIC die Daten für Zeile 0 vorfand. Dies muß leider in einer LDA-STA-Kette geschehen, da eine indizierte Schleife zu lange dauert. Der Bildschirm ist jetzt um 7 Bit nach unten verschoben, und der VIC sieht sich plötzlich mit der Tatsache konfrontiert, daß er jetzt zum zweiten Mal eine Bildschirmzeile ab $30 aufbauen muß. Also holt er die Daten zum Aufbau dieser Zeile zum zweiten Mal ab der Adresse $0400. Der ganze Trick besteht also darin, den Arbeitsspeicher ab $0400 doppelt zu benutzen und die Daten zum richtigen Zeitpunkt schnell genug auszutauschen.

Der Teil des Programms, der im Interrupt hängt, belegt den Speicherbereich von $C162 bis $ C2C7. Weil der Rand abgeschaltet ist, muß die Bildschirmfarbe dafür herhalten, wenn in $C160 Bit 7 nicht gesetzt ist. Dann wird die Bildschirmfarbe in $C161 zeitweilig zwischengespeichert und der Wert aus $C160 nach $D021 geschoben. Dadurch kann ein oberer und unterer Rand simuliert werden. Die Routinen zum Ein- und Ausschalten vom Interrupt sowie das Verwalten der Daten für Zeile 0 befinden sich im Bereich von $C000 bis $C15F. Sie enthalten keine programmtechnischen Besonderheiten, außer daß bei den Routinen zum Auswerten und Anzeigen eines Textes die Umwandlung von ASCII-Code in Bildschirmcode »von Hand« gemacht werden muß, da das Betriebssystem keine derartige Routine anbietet.

Es wurde streng darauf geachtet, daß unterhalb von $C000 kein Speicherplatz belegt wird, weder in der Zeropage noch sonstwo. Die Kompatibilität zu allen Basic-Programmen, die keine Daten im $C000-Bereich ablegen, ist damit gesichert. (B. Chevreux/ef)

Beispiele für die Benutzung der Befehle

```
 0 SYS 49152 : REM 26-ZEILENMODUS AN
10 A=17 : T$=" DEMO"   : SYS 49358,A,T$
10 C=SQR(2*)+SIN(0.865) : SYS 49369,C
10 SYS49369," DEMOTEXT"
10 SYS49358,10," ***** GESCHAFFT *****"
10 POKE1,53:PRINT "[CLR]" :REM
BILDSCHIRM UND ZEILE 26 LOESCHEN
10 POKE1,55:PRINT "[CLR]" :REM
BILDSCHIRM OHNE ZEILE 26 LOESCHEN
10 SYS 49168 : REM ZEILE NULL LOESCHEN
20 SYS 49179 : REM 26-ZEILENMODUS AUS
```

Programmbeschreibung zu »Generator 26«

Länge in Bytes:	711 ($2C7)
Benutzter Speicher:	49152 bis 49950 ($C000 bis $C327)
Programm:	$C000 bis $C2C7
Bufferspeicher für Zeile 1:	$C2D0 bis $C2F7
Datenspeicher für Zeile 0:	$C300 bis $C327

```
Name : generator 26       c000 c2c8
-----------------------------------
c000 : 20 10 c0 20 1b c0 20 3b  0b
c008 : c0 20 7d c0 20 a2 c0 60  2b
c010 : a2 27 a9 20 9d 00 c3 ca  33
c018 : 10 fa 60 78 a9 00 a2 31  54
c020 : a0 ea 8d 1a d0 8e 14 03  b4
c028 : 8c 15 03 a9 1b a2 40 a0  3e
c030 : 37 8d 11 d0 8e 05 dc 84  1a
c038 : 01 58 60 78 a2 61 a9 9f  a8
c040 : 8d 48 c0 8d 4b c0 b9 00  75
c048 : 00 99 00 00 c8 d0 f7 ee  e6
c050 : 48 c0 ee 4b c0 ca d0 ee  a1
c058 : 58 a9 35 85 01 a9 4c 8d  2d
c060 : 44 e5 8d ea e8 a9 c5 a2  90
c068 : c0 8d 45 e5 8e 46 e5 a9  03
c070 : b4 a2 c0 8d eb e8 8e ec  71
c078 : e8 20 dd fd 60 78 a9 99  4b
c080 : a2 c0 a0 81 8d 14 03 8e  7d
c088 : 15 03 8c 1a d0 a9 1b a2  91
c090 : f9 8d 11 d0 8e 12 d0 58  1c
c098 : 60 ad 19 d0 8d 19 d0 4c  ad
c0a0 : 31 ea 78 a9 62 a2 c1 a0  1d
c0a8 : 2f 8d 14 03 8e 15 03 8c  ba
c0b0 : 05 dc 58 60 a2 27 bd 00  a0
c0b8 : 04 9d 00 c3 ca 10 f7 a5  5c
c0c0 : ac 48 4c ed e8 20 10 c0  b2
c0c8 : ad 88 02 4c 47 e5 20 fd  e4
c0d0 : ae 20 9e b7 8a e0 27 90  9a
c0d8 : 02 a9 00 18 69 00 8d 37  ed
c0e0 : c1 20 fd ae 20 9e ad 24  fc
c0e8 : 0d 30 06 20 dd bd 20 87  ee
c0f0 : b4 20 a6 b6 aa a0 00 e8  b6
c0f8 : ca f0 42 b1 22 30 3f c9  35
c100 : 0d d0 03 4c 3d c1 c9 1d  03
c108 : f0 2f c9 12 d0 0a 85 c7  47
c110 : e6 22 d0 e7 e6 23 d0 e3  cb
c118 : c9 92 d0 04 a9 00 f0 ee  1b
c120 : c9 20 90 0a c9 60 90 04  49
c128 : 29 df d0 02 29 3f e6 c7  6d
c130 : c6 c7 f0 02 09 80 99 00  51
c138 : c3 c8 4c f8 c0 60 29 7f  44
c140 : c9 7f d0 02 a9 5e c9 12  16
c148 : f0 d2 c9 20 90 05 09 40  ee
c150 : 4c 2e c1 c9 0d d0 f9 f0  7e
c158 : e4 00 00 00 00 00 00 00  3d
c160 : 0e f6 ad 19 d0 8d 19 d0  f7
c168 : 30 07 ad 0d dc 58 4c 31  4d
c170 : ea ad 12 d0 c9 f9 d0 0d  99
```

**Listing 17.
»Generator 26« entlockt dem C64 eine Bildschirmzeile mehr**

```
c178 : a9 17 8d 11 d0 a9 ff 8d  a8
c180 : 12 d0 4c 7e ea ad 60 c1  fe
c188 : 30 0c ad 21 d0 8d 61 c1  d0
c190 : ad 60 c1 8d 21 d0 a9 f9  c3
c198 : 8d 12 d0 a9 18 8d 11 d0  6b
c1a0 : a2 27 bd 00 04 9d d0 c2  3b
c1a8 : bd 00 c3 9d 00 04 ca 10  75
c1b0 : f1 ad 12 d0 c9 2e d0 f9  5c
c1b8 : ad 11 d0 29 80 d0 f2 ad  fd
c1c0 : 60 c1 30 06 ad 61 c1 8d  d6
c1c8 : 21 d0 a2 17 ca d0 fd a9  5b
c1d0 : 1f 8d 11 d0 ad d0 c2 8d  9c
c1d8 : 00 04 ad d1 c2 8d 01 04  24
c1e0 : ad d2 c2 8d 02 04 ad d3  f7
c1e8 : c2 8d 03 04 ad d4 c2 8d  5a
c1f0 : 04 04 ad d5 c2 8d 05 04  d1
c1f8 : ad d6 c2 8d 06 04 ad d7  5a
c200 : c2 8d 07 04 ad d8 c2 8d  93
c208 : 08 04 ad d9 c2 8d 09 04  7e
c210 : ad da c2 8d 0a 04 ad db  bc
c218 : c2 8d 0b 04 ad dc c2 8d  cc
c220 : 0c 04 ad dd c2 8d 0d 04  2a
c228 : ad de c2 8d 0e 04 ad df  1e
c230 : c2 8d 0f 04 ad e0 c2 8d  05
c238 : 10 04 ad e1 c2 8d 11 04  d7
c240 : ad e2 c2 8d 12 04 ad e3  80
c248 : c2 8d 13 04 ad e4 c2 8d  3e
c250 : 14 04 ad e5 c2 8d 15 04  83
c258 : ad e6 c2 8d 16 04 ad e7  e3
c260 : c2 8d 17 04 ad e8 c2 8d  77
c268 : 18 04 ad e9 c2 8d 19 04  30
c270 : ad ea c2 8d 1a 04 ad eb  45
c278 : c2 8d 1b 04 ad ec c2 8d  b1
c280 : 1c 04 ad ed c2 8d 1d 04  dc
c288 : ad ee c2 8d 1e 04 ad ef  a7
c290 : c2 8d 1f 04 ad f0 c2 8d  ea
c298 : 20 04 ad f1 c2 8d 21 04  89
c2a0 : ad f2 c2 8d 22 04 ad f3  09
c2a8 : c2 8d 23 04 ad f4 c2 8d  23
c2b0 : 24 04 ad f5 c2 8d 25 04  36
c2b8 : ad f6 c2 8d 26 04 ad f7  6c
c2c0 : c2 8d 27 04 4c 7e ea 00  f8
```

Listing 17. (Schluß)

11. Der Doppel-VIC

Nach Hunderten von Sprites auf einmal auf dem Bildschirm - die nur leider nicht zu benutzen waren - wollte ich mit meinem Programm »Multi 16« (Listing 19, bitte mit dem MSE eingeben) etwas »back to the roots«. Es stellt zwar nur 16 Sprites auf dem Bildschirm dar, kann sie jedoch uneingeschränkt verwalten. Es erlaubt für jedes der 16 Sprites eine beliebige Position, Farbe, Multicolor und X/Y-Vergrößerung. Listing 18 zeigt ein Demoprogramm.

Multi 16 liegt ab Speicherzelle $CE00. Zum Start wählt man zwischen SYS 52992 (nur den IRQ starten) und SYS 52995 (auch den neuen VIC initialisieren). Mit SYS 52998 wird Multi 16 wieder ausgeschaltet und der VIC in seinen Ausgangszustand versetzt. Ab $CF00 finden Sie die Register für die Sprites 1 bis 8 und ab $CF30 die für Nummer 9 bis 16. Die Belegung entspricht genau der des Original-VIC. Es werden aber nur die Speicherstellen $D000 bis $D010, $D015, $D017, $D01b bis $D01f und $D025 bis $D02F berücksichtigt. Die übrigen VIC-Register verwenden Sie auch weiterhin ganz normal. Ab $CF60 und $CF70 müssen die Spritepointer stehen, die immer nach $07F8 verschoben werden. Den Bildschirm darf man also nicht an andere Adressen verlegen. Ansonsten ist alles erlaubt (wie z.B. geänderter Zeichensatz). Die Funktionsweise ist ähnlich der des 136-Farbendemos aus Ausgabe 3/88 des 64'er-Magazins. Auch hier bei Multi 16 wird der Trick mit dem Rasterinterrupt verwendet. Es wird einmal der VIC ab $CF00 und einmal der ab $CF30 gezeigt. Aus demselben Grund flimmern auch die Sprites etwas, vor allem bei hellen Farben. (F. Deinzer/ef)

```
10 REM **   MUTLI 16 WRITTEN '88 BY          <007>
11 REM  *         FRANK DEINZER              <154>
12 REM  *         TANNENSTRASSE 20           <197>
13 REM  *  8505 ROETHENBACH                  <218>
14 REM **                                    <063>
15 REM              DEMO                     <063>
16 :                                        <248>
18 IF PEEK(52736)<>76 THEN LOAD"MULTI 16",
   8,1                                      <167>
19 SYS 58648:SYS 64789:SYS 64931            <121>
20 POKE 53281,.:POKE 53280,.                <007>
23 FOR T=12864 TO 13952:POKE T,0:NEXT       <054>
25 BA=49152+15*256 : REM BASISADRESSE VON
   VIC-MULTI 16                             <208>
30 V1=BA:V2=BA+3*16: REM NEUEN VIC ADRESSE
   N                                        <016>
35 Z1=BA+6*16:Z2=BA+7*16:REM SPRITEPOINTER
    ADRESSEN                                <016>
40 SA=49152+14*256:REM STARTADRESSE         <185>
50 PRINT"{CLR,WHITE}DIESES PROGRAMM KANN 1
   6 SPRITES UN-"                           <209>
55 PRINT"EINGESCHRAENKT ANZEIGEN (AUCH Z.B
    ALLE"                                   <160>
60 PRINT"16 NEBENEINANDER)."                <074>
65 PRINT"EINEN MOMENT BITTE":POKE 56334,0:
   POKE 1,51:PRINT"{2DOWN}"                 <235>
70 FOR T=0 TO 15:PO=201*64+T*64             <023>
75 NU$=RIGHT$("0"+RIGHT$(STR$(T+1),LEN(STR
   $(T+1))-1),2)                            <187>
80 A1=ASC(LEFT$(NU$,1)):A2=ASC(RIGHT$(NU$,
   1)):PRINT"{UP}"NU$                       <178>
85 FOR R=0 TO 7                             <226>
90 POKE PO+R*3,PEEK(A1*8+53248+R)           <124>
95 POKE PO+R*3+1,PEEK(A2*8+53248+R)         <188>
100 NEXT R                                  <000>
105 NEXT T                                  <021>
110 POKE 1,55:POKE 56334,1                  <245>
113 SYS SA+3:REM START INCL. INIT           <142>
115 PRINT"{CLR}JETZT WERDEN NACHEINANDER D
    IE 16 SPRITE"                           <102>
120 PRINT"EINGESCHALTET:":FOR T=0 TO 7:POK
    E Z1+T,201+T:NEXT                       <232>
123 FOR T=0 TO 7:POKE Z2+T,209+T:NEXT       <180>
125 FOR T=1 TO 15 STEP 2:POKE V1+T,100:NEX
    T                                       <023>
130 A=20:FOR T=0 TO 14 STEP 2:POKE V1+T,A:
    A=A+20:NEXT                             <162>
135 FOR T=1 TO 15 STEP 2:POKE V2+T,100:NEX
    T                                       <097>
140 FOR T=0 TO 14 STEP 2:POKE V2+T,A:A=A+2
    0:IF A>255 THEN                         <197>
145 IF A>255 THEN A=A-256:POKE V2+16,240    <150>
150 NEXT                                    <160>
155 L=0:FOR T=0 TO 7:L=L+2↑T:POKE V1+21,L   <179>
160 FOR R=1 TO 1000:NEXT R:NEXT T           <000>
165 L=0:FOR T=0 TO 7:L=L+2↑T:POKE V2+21,L   <062>
170 FOR R=1 TO 1000:NEXT R:NEXT T           <010>
175 FOR T=1 TO 15 STEP 2:FOR R=100 TO 200
    STEP 1:POKE V1+T,INT(R):NEXT R,T        <058>
180 FOR T=1 TO 15 STEP 2:FOR R=100 TO 200
    STEP 1:POKE V2+T,INT(R):NEXT R,T        <071>
190 FOR T=39 TO 46:POKE V1+T,1:FOR R=1 TO
    300:NEXT R,T                            <160>
195 FOR T=39 TO 46:POKE V2+T,1:FOR R=1 TO
    300:NEXT R,T                            <197>
200 FOR T=39 TO 46:POKE V1+T,2:FOR R=1 TO
    300:NEXT R,T                            <172>
205 FOR T=39 TO 46:POKE V2+T,2:FOR R=1 TO
    300:NEXT R,T                            <209>
210 FOR T=1 TO 4000:NEXT:RUN                <064>
```

Listing 18. Ein Demo zu Multi 16

```
Name : multi 16          ce00 cef8
-----------------------------------
ce00 : 4c 29 ce 4c 09 ce 4c d8  08
ce08 : ce a0 00 b9 00 d0 99 00  4b
ce10 : cf 99 30 cf c8 c0 2f d0  a3
ce18 : f2 a0 00 b9 f8 07 99 60  80
ce20 : cf 99 70 cf c8 c0 08 d0  26
ce28 : f2 78 a0 00 b9 00 d0 99  91
```

Listing 19. ▶

```
ce30 : 80 cf c8 c0 30 d0 f5 a9  97
ce38 : 5a 8d 14 03 a9 ce 8d 15  30
ce40 : 03 a9 81 8d 1a d0 a9 01  fb
ce48 : 8d 12 d0 ad 11 d0 29 7f  03
ce50 : 8d 11 d0 a9 00 8d d6 ce  34
ce58 : 58 60 ad 19 d0 8d 19 d0  ee
ce60 : 30 07 ad 0d dc 58 4c 31  45
ce68 : ea ad d6 ce f0 05 a2 30  da
ce70 : 4c 75 ce a2 00 8e d7 ce  f0
ce78 : a0 00 bd 00 cf 99 00 d0  f3
ce80 : e8 c8 c0 11 d0 f4 ae d7  3e
ce88 : ce bd 15 cf 8d 15 d0 bd  b4
ce90 : 17 cf 8d 17 d0 a0 1b bd  cf
ce98 : 1b cf 99 00 d0 e8 c8 c0  fa
cea0 : 20 d0 f4 ae d7 ce a0 25  fc
cea8 : bd 25 cf 99 00 d0 e8 c8  db
ceb0 : c0 2f d0 f4 a0 f8 ae d7  17
ceb8 : ce e0 00 f0 02 a2 10 bd  05
cec0 : 60 cf 99 00 07 e8 c8 c0  cb
cec8 : 00 d0 f4 ad d6 ce 49 01  2e
ced0 : 8d d6 ce 4c 7e ea 00 00  45
ced8 : 78 a9 31 8d 14 03 a9 ea  f9
cee0 : 8d 15 03 a9 00 8d 1a d0  64
cee8 : a0 00 b9 80 cf 99 00 d0  72
cef0 : c8 c0 30 d0 f5 58 60 79  d5
```

Listing 19. 16 unabhängige Sprites auf einmal auf dem Bildschirm

12. »Flexible Line Distance«

Nach so vielen Tricks, die den Sprites oder den Farben des C64 unter die Arme greifen, wird es Zeit, den Bildschirm selbst einmal in Bewegung zu setzen. Wir wollen ihn jedoch nicht nur einfach horizontal oder vertikal scrollen; wir wollen jeder Bildschirmzeile einen variablen Abstand zur vorhergehenden Zeile geben. Auch hier heißt das Lösungswort: Raster-IRQs.

Nachdem man das Programm »F.L.D.« (Listing 20) mit dem MSE eingegeben und mit LOAD "F.L.D",8,1 geladen hat, ist es mit SYS 4096 zu starten. Eine IRQ-Routine ab $1027 wird initialisiert und von nun an jede 1/60 Sekunde aufgerufen.

Diese Routine hat die Aufgabe, die Arbeit des VIC zwischen zwei Bildschirmzeilen für kurze Zeit zu verzögern, um so einen bestimmten Abstand zwischen den Zeilen zu erzeugen. Zu diesem Zweck wird aus $D012 die aktuelle Rasterzeile gelesen, mit

AND #$07

auf einen Wert zwischen 0 und 7 reduziert und dann in $D011 abgelegt (dieses Register ist unter anderem für das bitweise vertikale Scrolling des C64 zuständig). Damit ist die aktuelle Rasterzeile um ein Pixel nach unten verschoben. Diese Prozedur wiederholt sich nun entsprechend der Anzahl der Rasterzeilen, die laut Tabelle zwischen der letzten und der aktuellen Bildschirmzeile freigelassen werden sollen. Anschließend verzögert eine Leerschleife die Arbeit der IRQ-Routine so lange, bis der Rasterstrahl acht Pixelzeilen auf den Bildschirm geschrieben hat, um dann für die Bearbeitung der nächsten Bildschirmzeile wieder an den Anfang der Routine zu verzweigen. In dieser Zeit wird die aktuelle Textzeile auf den Bildschirm gebracht. Zum Schluß der Routine wird wieder in den normalen System-Interrupt verzweigt.

Um nun den rollierenden Effekt zu erreichen, der sich nach dem Start der Routine auf dem Bildschirm zeigt, werden einfach für jede Zeile variable Abstandswerte gewählt, die zudem noch zyklisch verändert werden. »Fld.src« (Listing 21) zeigt, wie das Programm aufgebaut ist.

(Michael Wandel/ef)

```
Name : f.l.d.               1000 10c0
-------------------------------------
1000 : 78 a9 7f 8d 0d dc a9 f1  21
1008 : 8d 1a d0 a9 1b 8d 11 d0  10
1010 : a9 2e 8d 12 d0 a9 27 8d  88
1018 : 14 03 a9 10 8d 15 03 a9  fb
1020 : 01 85 03 85 02 58 60 a9  0d
1028 : 01 8d 19 d0 a4 02 a2 00  35
1030 : ad 12 d0 cd 12 d0 f0 fb  37
1038 : 29 07 69 18 8d 11 d0 e8  b9
1040 : 8a d9 80 10 d0 ea 98 48  30
1048 : 24 ea ea a2 00 ea a0 08  da
1050 : 88 10 fd e8 e0 07 d0 f5  f2
1058 : 68 a8 c8 e6 03 a5 03 c9  20
1060 : 10 d0 cb a9 00 85 03 e6  06
1068 : 02 a5 02 c9 20 d0 04 a9  e3
1070 : 00 85 02 4c 31 ea 00 00  a7
1078 : 00 00 00 00 00 00 00 00  79
1080 : 01 01 02 02 03 03 03 04  1f
1088 : 04 04 05 05 05 05 06 06  0d
1090 : 06 06 05 05 05 05 04 04  0c
1098 : 04 03 03 03 02 02 01 01  75
10a0 : 01 01 02 02 03 03 03 04  3f
10a8 : 04 04 05 05 05 05 06 06  2d
10b0 : 06 06 05 05 05 05 04 04  2c
10b8 : 04 03 03 03 02 02 01 01  95
```

Listing 20. Mit »F.L.D.« verfügt der Bildschirm des C64 über variable Zeilenabstände

```
-;
-.ba$1000
-;
-               sei             ; Interrupt verhindern
-               lda #$7f
-               sta $dc0d
-               lda #$f1
-               sta $d01a
-               lda #$1b
-               sta $d011
-               lda #$2e
-               sta $d012
-               lda #$27
-               sta $0314
-               lda #$10
-               sta $0315       ; IRQ-Routine initialisieren
-               lda #$01
-               sta $03
-               sta $02         ; Laufvariablen initialisieren
-               cli
-               rts
-;
-               lda #$01
-               sta $d019       ; IRQ-Flag loeschen
-               ldy $02
-line           ldx #$00
-dist           lda $d012
-loop1          cmp $d012       ; noch gleiche Zeile?
-               beq loop1       ; wenn nein, dann weiter
-               and #$07
-               adc #$18
-               sta $d011       ; verknuepfen und in $D011 ablegen
-               inx
-               txa
-               cmp table,y     ; Vergleich mit Tabelle
-               bne dist        ; wenn nein, dann zurueck
-               tya
-               pha             ; Y retten
-               bit $ea
-               nop
-               ldx #$00
-loop2          nop
-               ldy #$08
-loop3          dey
-               bpl loop3
-               inx
-               cpx #$07
-               bne loop2       ; Verzoegerung um 8 Pixelzeilen
-               pla
-               tay             ; Y zurueckholen
-               iny
-               inc $03
-               lda $03
-               cmp #$10        ; Anzahl der Durchlaeufe/Zeilen erreicht?
-               bne line
-               lda #$00
-               sta $03
-               inc $02
-               lda $02         ; wenn ja, dann Tabellenzeiger um 1 erhoehen
-               cmp #$20        ; schon alle Werte?
-               bne end
-               lda #$00
-               sta $02
-end            jmp $ea31       ; wenn ja, dann zu System-IRQ verzweigen
-;
-;-----------------------------------------
-;
-table          .by $01,$01,$02,$02,$03,$03,$03,$04
-               .by $04,$04,$05,$05,$05,$05,$06,$06
-               .by $06,$06,$05,$05,$05,$05,$04,$04
-               .by $04,$03,$03,$03,$02,$02,$01,$01
-               .by $01,$01,$02,$02,$03,$03,$03,$04
-               .by $04,$04,$05,$05,$05,$05,$06,$06
-               .by $06,$06,$05,$05,$05,$05,$04,$04
-               .by $04,$03,$03,$03,$02,$02,$01,$01
```

© 64'er

Listing 21. Das dokumentierte Source-Listing zu »F.L.D.«

13. Paint Magic und Basic-Programme

Um Paint Magic-Bilder in einem Basic-Programm aufrufen zu können, geht man wie folgt vor:
- Laden des Bildes.
- Starten des Bildes mit »RUN«.
- <RUN/STOP RESTORE> drücken.
- Im Direktmodus »POKE 24565,96« eingeben.
- Speichern mit einem Monitor (zum Beispiel SMON) von Hex $4000 bis $6400.

Das Bild kann jetzt, wenn es zuvor absolut geladen wurde, in einem Basic-Programm mit »SYS 24513« aufgerufen werden. Bei Problemen können Sie als Alternativen statt POKE 24565,96 auch POKE 24565,60 oder POKE 24565,68 eingeben. Statt SYS24513 kann das Bild eventuell mit SYS24518 aufgerufen werden.

Verschwinden des Bildes:

```
POKE 53272,21
POKE 56576,151
POKE 53265,27
POKE 53270,200
```

(Frank Hoffmann/ef)

14. Holzauge sei wachsam

Wer kennt das nicht: Ein Programm gibt keinen Mucks mehr von sich. Ist es jetzt abgestürzt oder hängt es in irgendeiner Berechnungsschleife?

Das Programm »Freemem« (Listing 22) zeigt interruptgesteuert den Stackpointer und den freien Basic-Speicher in der rechten oberen Bildschirmecke. Vor allem die Anzeige des Stackpointers, die bei einem normalen Programmlauf ständig wechselt, ist eine tolle Einrichtung zum Austesten von kritischen Programmen. Auch können Sie mit Hilfe der Anzeige des freien Speichers nun jederzeit feststellen, wann der Computer gerade eine Garbage-Collection durchführt.

»Freemem« wird mit SYS 53100 gestartet und mit <RUN/STOP RESTORE> wieder abgeschaltet.

(Henning Müller-Zauleck/ef)

Listing 22. »Freemem«

```
Name : freemem 53100      cf6c cfc7
-------------------------------------
cf6c : 78 ad 14 03 8d a5 cf ad  c1
cf74 : 15 03 8d a6 cf a9 85 8d  be
cf7c : 14 03 a9 cf 8d 15 03 58  b4
cf84 : 60 78 ba 8a a2 ff 20 a7  1a
cf8c : cf a9 20 e8 9d 21 04 38  b8
cf94 : a5 33 e5 31 a8 a5 34 e5  c7
cf9c : 32 20 a7 cf 98 20 a7 cf  8b
cfa4 : 4c 00 00 48 4a 4a 4a 4a  ae
cfac : e8 20 b8 cf 68 29 0f e8  aa
cfb4 : 20 b8 cf 60 c9 0a b0 03  e6
cfbc : 69 30 2c e9 09 09 80 9d  9c
cfc4 : 21 04 60 00 00 00 00 00  00
```

15. Übersichtliche Programme

Das Basic des C64 hat einen Nachteil: Sie können nicht, wie in anderen Programmiersprachen üblich, Basic-Zeilen durch Einrücken übersichtlicher gestalten.

Ein Beispiel:

```
10 REM SO WAERE ES DOCH SCHOEN, ODER?
20 PRINT CHR$(147)
30 FOR Y=1 TO 5
40    FOR X=1 TO 10
50       PRINT X,Y
60    NEXT X
70 NEXT Y
```

Durch die Einrückungen erhöht sich natürlich die Lesbarkeit des Programms, vor allem innerhalb von Schleifen. Wenn Sie aber das Listing genauso wie es oben steht eintippen, ignoriert der Computer leider die Leerzeichen in den Zeilen 40 bis 60. Durch einen Trick läßt sich dies umgehen: Tippen Sie »POKE 131,0«, bevor (!) Sie mit dem Eingeben des Programms beginnen. Dadurch wird eine Maschinenroutine im Speicher Ihres C64 verändert, die normalerweise alle Leerzeichen am Anfang von Basic-Zeilen ignoriert.

Aber Vorsicht: Bevor Sie ein auf diese Weise geändertes Programm starten können, müssen Sie »POKE 131,239« eingeben. Dadurch wird die Änderung in der oben erwähnten Maschinenroutine wieder rückgängig gemacht.

(Ralf Habermann/ef)

Aber es gibt noch andere Methoden, diese Strukturierung auch ohne den genannten POKE-Befehl zu erhalten:

1. Der Trick mit den Sonderzeichen

Wenn das erste Zeichen in einer Basic-Zeile irgendein Sonderzeichen (zum Beispiel <SHIFT A>) ist, werden alle nachfolgenden Leerzeichen korrekt angenommen. Das Programm ist dann trotzdem lauffähig, und man kommt ohne POKEs aus. Sie tippen also die Zeile 30 so ein:

```
30 {SHIFT A}{5SPACE}FOR X=1 TO 10
```

und nach dem Eingeben von LIST erscheint:

```
30          FOR X=1 TO 10
```

Erklärung: Der C64 übergeht beim Eingeben von Basic-Programmen alle Sonderzeichen, die nicht innerhalb von Anführungszeichen stehen. Normalerweise würde er auch die nachfolgenden Spaces übergehen, was aber aufgrund einer kleinen Ungenauigkeit in der zuständigen Maschinenroutine nicht der Fall ist. Fazit: Die Leerzeichen am Anfang der Zeile bleiben bestehen und tragen zur besseren Lesbarkeit des Programms bei. (Dietmar Grabs/ef)

2. Ganz einfach geht's mit dem Doppelpunkt

Und wem die oben genannte Methode noch zu umständlich ist, der verwendet einfach einen Doppelpunkt am Anfang der jeweiligen Zeile. Das könnte dann so aussehen:

```
10 REM BEISPIEL-PROGRAMM
20 FOR Y=1 TO 5
30 :          FOR X=1 TO 10
40 :                  PRINT X,Y
50 :          NEXT X
60 NEXT Y
```

Dieser Trick hat den Vorteil, daß Sie in mit dem Doppelpunkt formatierten Zeilen auch noch Änderungen machen können (=editieren). Auch Leerzeilen ohne Inhalt zur strukturierten Programmierung lassen sich damit erzeugen. Beispiel:

```
.
.
.
170 END
180 REM ENDE HAUPTPROGRAMM
190 :
200 REM ANFANG UNTERPROGRAMM 1
210 A=B*10
.
.
.
```

(Michael Kammer/Peter Gorgs/ef)

16. Schnelleres Basic mit Quickjump

Geht Ihnen nicht auch manchmal der langsame Basic-Interpreter auf die Nerven? Quickjump macht Ihre Basic-Programme bis zu fünfmal schneller – und das ohne Compiler.

Beim Bearbeiten eines Basic-Programms holt sich der Interpreter die zu bearbeitende Zeile aus dem Basic-Text und speichert sie in einen speziellen Puffer. Hier wird die Zeile Befehl für Befehl ausgeführt. Bei jedem Befehlswort muß der Interpreter in einer Tabelle die entsprechende Einsprungadresse im Betriebssystem suchen, was der erste Grund dafür ist, daß die Basic-Programme relativ langsam sind. Der zweite Grund ist der, daß bei jedem GOTO- oder GOSUB-Befehl die entsprechende Zeilennummer im

```
Name : quickjump           c000 c0ec
-------------------------------------
c000 : 4c 06 c0 00 00 d0 a0 00   89
c008 : 84 22 a9 a0 85 23 b1 22   98
c010 : 91 22 c8 d0 f9 e6 23 a9   b5
c018 : c0 c5 23 d0 f1 a9 42 8d   2e
c020 : c1 a8 a9 c0 8d c2 a8 a9   9d
c028 : e0 8d 20 a0 a9 c0 8d 21   04
c030 : a0 a5 01 29 fe 85 01 a9   7c
c038 : ea 8d e8 c0 a9 c0 8d e9   e6
c040 : c0 60 85 5f 86 60 a9 c0   11
c048 : cd e9 c0 d0 09 a9 ea cd   79
c050 : e8 c0 d0 02 f0 3e a9 ea   8a
c058 : 85 22 a9 c0 85 23 a0 00   65
c060 : b1 22 c5 14 d0 07 c8 b1   e2
c068 : 22 c5 15 f0 1b a5 22 18   68
c070 : 69 04 85 22 a5 23 69 00   9a
c078 : 85 23 cd e9 c0 d0 df a5   9d
c080 : 22 cd e8 c0 d0 d8 f0 0c   8b
c088 : c8 b1 22 85 5f c8 b1 22   a9
c090 : 85 60 38 60 a0 01 20 1d   2c
c098 : a6 b0 02 18 60 ad 03 c0   1b
c0a0 : d0 3c ad e9 c0 cd 05 c0   47
c0a8 : d0 0a ad e8 c0 cd 04 c0   12
c0b0 : d0 02 38 60 a0 00 ad e8   2e
c0b8 : c0 85 22 18 69 04 8d e8   85
c0c0 : c0 ad e9 c0 85 23 69 00   00
c0c8 : 8d e9 c0 a5 14 91 22 c8   17
c0d0 : a5 15 91 22 c8 a5 5f 91   03
c0d8 : 22 c8 a5 60 91 22 38 60   a0
c0e0 : 00 08 20 37 c0 4c 72 a8   5d
c0e8 : 00 00 00 17 dd bd 20 87   27
```

Listing 23. »Quickjump« macht Ihren Basic-Programmen Beine. Es ist mit dem MSE (siehe Seite 159) einzugeben und zu speichern. Gestartet wird »Quickjump« mit LOAD-"QUICKJUMP",8,1:SYS 49152 <RETURN>.

```
         .opt oo,p4
vektor   =    $22
         *=   $c000
start    jmp  copy         ; sprung zur initialisierung
flag     .byt 0           ; flag fuer tabelleneintragung
tabend   .byt $00,$d0     ; tabellenende
 ; ******************
 ; **initialisierung**
 ; ******************
 ; *** betriebssystem kopieren
copy     ldy  #0
         sty  vektor
         lda  #$a0
         sta  vektor+1
loop1    lda  (vektor),y
         sta  (vektor),y
         iny
         bne  loop1
         inc  vektor+1
         lda  #$c0
         cmp  vektor+1
         bne  loop1
 ; *** sprung in eigene zeilensuchroutine einbinden
         lda  #<tab
         sta  $a8c1
         lda  #>tab
         sta  $a8c2
 ; *** tabelle loeschen bei 'run' einbinden
         lda  #<beg
         sta  $a020
         lda  #>beg
         sta  $a021
 ; *** basic-rom ausblenden
         lda  $1
         and  #254
         sta  $1
 ; *** tabelle loeschen
init     lda  #<tabanf
         sta  tabptr
         lda  #>tabanf
         sta  tabptr+1
         rts
 ; ****************************
 ; **eigene zeilensuchroutine**
 ; ****************************
tab      sta  $5f
         stx  $60                  ; zn ab der zu suchen ist
 ; ****************************
 ; *** test ob tabelle leer ***
 ; ****************************
         lda  #>tabanf
         cmp  tabptr+1
         bne  noemp
         lda  #<tabanf
         cmp  tabptr
         bne  noemp
         beq  srch
 ; ***************************
 ; *** tabelle durchsuchen ***
 ; ***************************
 ; zeiger an tabellenanfang
noemp    lda  #<tabanf
         sta  vektor
         lda  #>tabanf
         sta  vektor+1
 ; *** mit gesuchter nummer vergleichen
l01      ldy  #0
         lda  (vektor),y
         cmp  $14
         bne  c01
         iny
         lda  (vektor),y
         cmp  $15
         beq  found
 ; *** zeiger auf naechsten eintrag
c01      lda  vektor
         clc
         adc  #4
         sta  vektor
         lda  vektor+1
         adc  #0
         sta  vektor+1
 ; *** wenn tabellenende noch nicht erreicht weitersuchen
         cmp  tabptr+1
         bne  l01
         lda  vektor
         cmp  tabptr
         bne  l01
 ; *** gesuchte nummer nicht in tabelle

         beq  srch
 ; *** zeilennummer gefunden - uebergeben
found    iny
         lda  (vektor),y
         sta  $5f
         iny
         lda  (vektor),y
         sta  $60
         sec
         rts
 ; ****************************
 ; *** programm durchsuchen ***
 ; ****************************
srch     ldy  #1
         jsr  $a61d
         bcs  fnd
 ; *****************************
 ; *** zeile existiert nicht ***
 ; *****************************
nfnd     clc
         rts
 ; **********************
 ; *** zeile gefunden ***
 ; **********************
 ; *** eintragung nicht gewuenscht
fnd      lda  flag
         bne  c02
 ; *** tabelle schon voll
         lda  tabptr+1
         cmp  tabend+1
```

Listing 24. Für all jene, die es genau wissen wollen: der Quelltext zu »Quickjump« ▶

```
            bne   apnd
            lda   tabptr
            cmp   tabend
            bne   apnd
            sec
            rts
    ; ***********************************
    ; *** nummer in tabelle eintragen ***
    ; ***********************************
    ; *** tabellenende-zeiger erhoehen
apnd        ldy   #0
            lda   tabptr
            sta   vektor
            clc
            adc   #4
            sta   tabptr
            lda   tabptr+1
            sta   vektor+1
            adc   #0
            sta   tabptr+1
    ; *** adresse in tabelle eintragen
            lda   $14
            sta   (vektor),y
            iny
            lda   $15
            sta   (vektor),y
            iny
            lda   $5f
            sta   (vektor),y
            iny
            lda   $60
            sta   (vektor),y
c02         sec
            rts
    ; ******************************
    ; **modifizierter 'run'-befehl**
    ; ******************************
beg         .byt 0
            php
            jsr  init
            jmp  $a872
    ; ************************************
    ; **tabellenzeiger und tabellenanfang**
    ; ************************************
tabptr      .byt 0,0
tabanf      .byt 0
```

Listing 24. Der Quelltext zu »Quickjump« (Schluß)

Basic-Text gesucht und die Verzweigungsadresse errechnet werden muß. Da sich der Interpreter diese Verzweigungsadresse nicht merkt, muß bei einem erneuten Aufruf die Zeilennummer wieder gesucht und die Verzweigungsadresse neu berechnet werden und das kostet Zeit. Genau an dieser Stelle greift zum Beispiel der Austro-Compiler an. Beim Übersetzen wird jede Zeilennummer durch eine absolute Adresse ersetzt. Quickjump (Listing 23) arbeitet ähnlich. Allerdings ist hier ein Compilerlauf überflüssig. Beim Bearbeiten des Programms legt Quickjump eine Tabelle an. Immer dann, wenn der Interpreter auf einen GOTO- oder GOSUB-Befehl stößt, wird zunächst in der Tabelle nachgesehen, ob die Verzweigungsadresse schon existiert. Existiert sie, verzweigt Quickjump ohne den Basic-Text zu durchsuchen. Existiert sie nicht, wird die Zeilennummer ganz normal gesucht, die Verzweigungsadresse errechnet und an die Tabelle angehängt.

Berechnet Zieladressen

Ein Test mit einem Quicksort-Unterprogramm ergab das in Tabelle 1 dargestellte Ergebnis. Es macht die Effizienz der Routine deutlich.

Quickjump belegt den Speicherbereich von $C000 bis $C0EB. Im Anschluß daran wird die Tabelle angelegt und reicht bis $CFFF.

Nach der Eingabe mit dem MSE wird das Programm mit LOAD"QUICKJUMP",8,1 geladen und mit SYS 49152 gestartet. Zuvor sollte man jedoch NEW <RETURN> eingeben, um die Basic-Zeiger zu initialisieren.

Programm am Anfang nach 300 REM-Zeilen			
Elemente:	50	50	100
normal:	15s	38s	146s
mit Quickjump:	14s	15s	56s

Tabelle 1. Geschwindigkeitsvergleich zwischen einem Quicksort-Unterprogramm mit und ohne »Quickjump«

Nach einem Reset läßt sich Quickjump durch POKE 1,54 erneut aktivieren.

Die Speicherzelle 49155 enthält ein Flag für die Aufnahme weiterer Adressen in die Tabelle. Steht dort 0 (standard), so werden neue Adressen eingetragen. Bei jeder anderen Zahl wird die Liste lediglich durchsucht. Neue Adressen werden aber nicht hinzugefügt. Dies kann man zur Vermeidung wenig relevanter Einträge benutzen.

Die Speicherzellen 49156 und 49157 enthalten Low- und Highbyte der Tabellenendadresse. Sie kann beliebig geändert werden (Standard ist $D000).

All jene, die Quickjump in ihre eigenen Programme einbauen oder genau wissen wollen, wie Quickjump funktioniert, finden in Listing 24 den dokumentierten Quellcode. Assembliert wurde das Programm mit Profi-Ass von Data-Becker. Es läßt sich jedoch nach Änderung der Pseudo-Opcodes jeder beliebige Assembler, so zum Beispiel das Programm Hypra-Ass aus Sonderheft 35 einsetzen.

(Detlef Keiler/ef)

17. Joystick glasklar

Durch diese kleine Maschinenroutine (siehe Listing 25) wird ein an den Computer angeschlossener Joystick abgefragt und der Schaltzustand in einer Basic-Variablen (!) abgelegt.

Der Aufruf erfolgt über »SYS 49152,Port-Nummer«, wobei für Port-Nummer entweder 1 oder 2 angegeben werden kann. Das Maschinenprogramm legt automatisch zwei Basic-Variablen an. Je nach gewählter Port-Nummer lauten diese J1/J2 und F1/F2. Dabei enthält J1/J2 nach dem Aufruf die Position des Joysticks:

```
5 1 6
3 0 4
7 2 8
```

Die Variable F1/F2 enthält bei gedrücktem Feuerknopf den Wert 1, ansonsten 0.

```
Name : joystick                c000 c073
-----------------------------------------
c000 : 20 fd ae 20 9e b7 ca e0  63
c008 : 02 90 05 a2 0e 4c 37 a4  51
c010 : e8 a9 31 e0 02 d0 04 a2  31
c018 : 00 a9 32 8d 6f c0 a9 4a  63
c020 : 8d 6e c0 bd 00 dc 48 29  26
c028 : 0f a0 08 d9 65 c0 f0 03  eb
c030 : 88 10 f8 98 20 47 c0 a9  a4
c038 : 46 8d 6e c0 68 29 10 c9  9c
c040 : 00 f0 02 a9 01 49 01 18  fd
c048 : 69 30 8d 71 c0 a5 7a 48  0f
c050 : a5 7b 48 a9 6e 85 7a a9  4a
c058 : c0 85 7b 20 a5 a9 68 85  12
c060 : 7b 68 85 7a 60 0f 0e 0d  91
c068 : 0b 07 0a 06 09 05 4a 31  7f
c070 : b2 30 00 22 a5 23 69 00  98
```

Listing 25. Geniale Joystick-Abfrage in Maschinensprache

Die Auswertung von J1/J2 könnte in Basic zum Beispiel so erfolgen:

SYS 49152,1:ON J1 GOTO ...

In Listing 26 sehen Sie den dokumentierten Quelltext zum Programm. Er wurde mit dem Assembler »Profi-Ass« geschrieben. (Andreas Wendker/ef)

```
        .opt p4
        *=   $c000
;
;  *** ma.joy ***
;
; andreas wendker
; gojenbergsweg 112 d
; 2050 hamburg 80
; tel. (040) 720 68 04
;
;
; aufruf -> sys 49152, nummer
;
; funktion -> joystick-abfrage
;
;
;

getbyte  =    $b79e   ;zahl (0-255) aus basic-text holen
 ;und in x uebergeben
chkcom   =    $aefd   ;auf komma pruefen
fehaus   =    $a437   ;fehlermeldung ausgeben
letvar   =    $a9a5   ;basic-befehl let bearbeiten
 ;

 ;joystick-port-nummer aus basic-text holen,
 ;werteberichs-ueberpruefung
 ;sowie bestimmung einiger parameter
 ;
start     jsr  chkcom  ;komma ueberspringen
          jsr  getbyte ;nummer des joystick-ports holen
          dex          ;pruefen, ob zahl = 1 bzw. 2
          cpx  #2
          bcc  ok
          ldx  #14     ;sonst fehlermeldung ausgeben
          jmp  fehaus  ;-> illegal quantity error

ok        inx          ;je nach joystick-port-nummer
          lda  #"1"    ;a mit "1" bzw. "2" laden und
          cpx  #2      ;x auf joystick-register
          bne  jonum   ;zeigen lassen
          ldx  #0
          lda  #"2"    ;port-nummer als 2. buchstaben
jonum     sta  text+1  ;des variablen-namens uebernehmen
 ;

 ;variabe 'j1' bzw. 'j2' bearbeiten
 ;-> je nach bewegungsrichtung
 ;des joysticks einen wert von 0 bis 8 zuweisen
 ;
          lda  #"j"    ;1.buchstabe = "j"
          sta  text    ;-> variable heisst nun j1 bzw. j2
          lda  56320,x ;wert des joystick-registers holen
          pha          ;und retten
          and  #%00001111 ;untere vier bits isolieren
          ldy  #8

josuch    cmp  verwer,y ;und mit allen werten vergleichen
          beq  found
          dey
          bpl  josuch
found     tya          ;a enthaelt jetzt den wert,
;der der variablen zugewiesen werden soll
          jsr  anlegen ;variable anlegen
;
;jetzt die variable 'f1' bzw. 'f2' bearbeiten
;-> je nach zustand des feuerknopfes 0 oder 1 zuweisen
;
          lda  #"f"    ;variablen-namen ab text umbenennen
          sta  text    ;in 'f1' bzw. 'f2'
          pla          ;joystick-register zurueckholen
          and  #%00010000 ;feuerknopf-bit isolieren
          cmp  #0
          beq  feuok   ;knopf gedrueckt
          lda  #1
feuok     eor  #1      ;bit 0 umdrehen, a enhaelt nun ergebn

;
;routine zum anlegen der variablen
;-> bei j1/j2 wird sie als unterprogramm aufgerufen,
;bei f1/f2 ganz normal durchgegangen
;-> in a muss der spaetere wert der variablen stehen
;
anlegen   clc          ;ascii-code der zahl ermitteln
          adc  #48
          sta  text+3  ;in basic-befehl ab text einfuegen
          lda  $7a     ;chrget-zeiger retten
          pha
          lda  $7b
          pha
          lda  #<text  ;chrget-zeiger auf eigenen befehl
          sta  $7a     ;ab text richten
          lda  #>text
          sta  $7b
          jsr  letvar  ;befehl abarbeiten -> wertzuweisung
          pla          ;chrget-zeiger auf altes byte richten
          sta  $7b
          pla
          sta  $7a
          rts

;
;alle denkbaren kombinationen der vier bewegungsrichtungen
;des joysticks -> reihenfolge entspricht
;den spaeteren variablen-werten
;

verwer    .byte%00001111 ;nicht bewegt
          .byte%00001110 ;oben
          .byte%00001101 ;unten
          .byte%00001011 ;links
          .byte%00000111 ;rechts
          .byte%00001010 ;oben-links
          .byte%00000110 ;oben-rechts
          .byte%00001001 ;unten-links
          .byte%00000101 ;unten-rechts
;

;basic-befehl    j1=0
;-> verschluesselt zur bearbeitung
;durch das betriebssystem
;-> befehl wird je nach variable geaendert
;
text      .asc "j1"     ;platz fuer variablen-namen
          .byte$b2      ;basic-token fuer '='
          .byte48, 0    ;platz fuer ergebnis-ascii-code
;die 0 kennzeichnet das befehls-ende
```

Listing 26.
Der Quelltext zur Joystick-Abfrage

18. Wie beim Atari ST – Varptr

Mit Hilfe dieses Tricks haben Sie die Möglichkeit, die Speicherposition von Variablen festzustellen. In einigen Basic-Dialekten ist eine solche Routine unter dem Namen VARPTR implementiert. Unser C64 kann das auch, man muß es ihm nur entlocken. Zunächst ein paar Vorinformationen zu der Art und Weise, wie der C64 Variablentypen kennzeichnet.

Variablen werden unabhängig vom jeweiligen Typ immer in 7 Byte gespeichert. In den beiden ersten Byte befinden sich die beiden ersten Zeichen des Variablennamens (weshalb dann auch nur diese beiden Zeichen zur Unterscheidung herangezogen werden), die restlichen 5 Byte enthalten je nach Variablentyp unterschiedliche Informationen. In der Tabelle ist das verdeutlicht.

Nun muß das Betriebssytem natürlich eine Möglichkeit haben, die Typen von Variablen zu unterscheiden, damit nichts Unvorhergesehenes mit ihnen geschieht. Variablennamen dürfen ja nur aus ungeshifteten Buchstaben und Ziffern bestehen, es werden also in den beiden Byte des Variablennamens nicht alle Bits benötigt. Zur Typbestimmung zieht der C64 jetzt die beiden höchstwertigen Bits heran und begutachtet ihre Konstellation innerhalb der beiden Byte. Ist keins der höchstwertigen Bits gesetzt, so handelt es sich um eine Real-Variable, sind beide 1, dann folgt ein Integerwert. Die Anordnung können Sie in der Tabelle nachlesen.

Hier nun der Trick:

```
2000  POKE 180, n1 OR s1: POKE 181,n2 OR s2
2010  POKE  69,PEEK(180): POKE  70,PEEK(181):
      SYS 45287
2020  ad= PEEK(780)+256*PEEK(782)
2030  RETURN
```

In N1 und N2 werden die ASCII-Werte der beiden Namenszeichen übergeben. Sollte kein zweites Zeichen existieren, erhält N2 den Wert 0. S1 und S2 enthalten entweder

0 (dann ist das höchste Bit nicht gesetzt) oder 128 (höchstes Bit auf 1). Die Zeile 2000 verschmilzt diese Vorgaben und schreibt sie an eine Stelle, an der sie normalerweise vor der Gefahr sicher sein können, vom Betriebssystem geändert zu werden. Die Zeile 2010 überträgt diese Werte dann in die Speicherstellen, die der Computer benutzt, um Variable ausfindig zu machen, weshalb in dieser Zeile natürlich keine Variablen mehr verwendet werden dürfen! Der SYS-Befehl führt in eine Systemroutine, die Variablen sucht oder neu anlegt. Sie liefert im Akku und im Y-Register des Prozessors die ermittelte Variablenadresse zurück, die in Zeile 2020 an AD übergeben wird. Damit ist VARPTR für den C 64 realisiert. (Arndt Dettke/ef)

Dieser Trick läßt sich jedoch noch vereinfachen:
Die Speicherstellen 71/72 enthalten die Adresse der aktuellen Variable. Um nun die Startadresse einer beliebigen Variable (hier z. B. A$) zu berechnen, geht man folgendermaßen vor:

```
2000 A$=A$A
2010 POKE 180,PEEK (71): POKE 181,PEEk (72)
2020 AD=PEEK (180)+256*PEEK (181)
```

In Zeile 2010 wird die Variable A$ aktualisiert, so daß ihre Startadresse vom Betriebssystem in die Speicherstellen 71/72 eingetragen wird. Anschließend wird der Inhalt dieser Speicherstellen nach 180/181 kopiert und schließlich in die Variable AD übernommen. (Markus Hammer/ef)

Typ	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Real	Erstes Namens-zeichen	Zweites Namens-zeichen	Exponent	Mantisse 1	Mantisse 2	Mantisse 3	Mantisse 4
Integer	Erstes Namens-zeichen	Zweites Namens-zeichen	Wert Lo-Byte	Wert Hi-Byte	unbenutzt	unbenutzt	unbenutzt
String	Erstes Namens-zeichen	Zweites Namens-zeichen	String-länge	Adresse Lo-Byte	Adresse Hi-Byte	unbenutzt	unbenutzt
FN	Erstes Namens-zeichen	Zweites Namens-zeichen	Adresse des Aus-drucks Lo	Adresse des Aus-drucks Hi	Adresse des Platz-halters Lo	Adresse des Platz-halters Hi	Erstes Zeichen nach '='

Organisation der Variablentypen im Speicher

19. Kurz und effektiv – PRINT AT

Sicher werden einige von Ihnen einwenden, daß man eine PRINT AT-Routine auch in Basic programmieren kann. Da haben Sie in gewisser Weise auch recht. Aber kann man in Basic noch von kurz und benutzerfreundlich sprechen? Diese andauernden Unterprogrammaufrufe verbrauchen auf die Dauer auch sehr viel Speicher (vom Unterprogramm selbst ganz zu schweigen).
Also muß ein Assembler her!
Da das Betriebssystem des C 64 einige hilfreiche Routinen zum Programmieren bietet, dürfte eine solche PRINT AT-Routine nicht schwer zu erstellen sein. Die Syntax soll dabei wie folgt aussehen:
SYS 49152, Spalte, Zeile, "text"
Hier vorab eine Liste der erforderlichen Betriebssystemroutinen:

```
$AAA0: PRINT-Routine des Betriebssystems
$AEFD: Prüft auf Komma hinter dem SYS-Befehl
$B7EB: Holt die Zahlen vor und hinter dem zweiten
       Komma
$FFF0: Positioniert den Cursor
```

Zunächst muß überprüft werden, ob hinter dem SYS-Befehl ein Komma steht. Das geschieht mit JSR $AEFD. Nun sind die beiden Positionsangaben (für Zeile und Spalte) aus der Befehlszeile zu holen. Dazu ist die Befehlsfolge »JSR $B7EB (Zahlen holen), TAY« erforderlich. Nun sind die Zahlen in den richtigen Registern, und der Cursor kann mit JMP $FFF0 positioniert werden. Da ein drittes Komma folgen soll, hinter dem der auszugebende Text steht, ist noch einmal die Routine zum Prüfen auf das Komma aufzurufen (JSR $AEFD). JMP $AAA0 verzweigt schließlich in die PRINT-Routine des Betriebssystems und gibt den angegebenen Text auf dem Bildschirm aus. Sollte man höhere Werte als erlaubt eingeben (Spalte > 39 oder Zeile > 24), stürzt der C 64 ab. Um das zu vermeiden, wurden in das Programm noch zwei Vergleichsbefehle, zwei Sprungbefehle und natürlich ein Unterprogramm, in das im Ernstfall verzweigt wird, eingebaut. Das Unterprogramm gibt bei falschen Parametern einen »illegal quantity error« aus. Um die Fehlermeldung auszugeben, wird die entsprechende Betriebssystem-Routine mit JMP $B248 angesprungen.
In Listing 27 finden Sie den Quellcode zum PRINT AT-Befehl. Bei Listing 28 handelt es sich um das lauffähige Programm. Es ist mit dem MSE einzugeben.
(Silvan Reinhold/ef)

```
100  -.OB"PRINT-AT.P.W"
150  -.BA $C000
200  -       JSR $AEFD       ;AUF KOMMA PRUEFEN
210  -       JSR $B7EB       ;ZAHLEN HOLEN
220  -       CPX #25         ;WENN X-REG. > ODER 25, DANN
230  -       BCS ERRO        ;JMP ERRROR
240  -       LDA $14
250  -       CMP #40         ;WENN AKKU > ODER = 40, DANN
260  -       BCS ERRO        ;JMP ERROR
270  -       TAY
280  -       JSR $FFF0       ;CURSOR SETZEN
300  -       JSR $AEFD       ;KOMMA TESTEN
310  -       JMP $AAA0       ;TEXT AUSGEBEN
320  -ERRO   JMP $B248       ;ILLEGAL QUANTITY
330  -.EN
```

© 64'er

Listing 27. Quellcode zum PRINT AT-Befehl

Listing 28. »PRINT AT« – kurz und komfortabel

```
Name : print-at            c000 c01d
-----------------------------------
c000 : 20 fd ae 20 eb b7 e0 19  01
c008 : b0 10 a5 14 c9 28 b0 0a  61
c010 : a8 20 f0 ff 20 fd ae 4c  4a
c018 : a0 aa 4c 48 b2 c0 a9 4a  96
```

20. Der C 64 spricht deutsch

So ganz ernst ist Listing 29 (»GERMAN BASIC«) nicht gemeint. Es handelt sich um ein Maschinenprogramm, das alle Basic-Wörter und Fehlermeldungen etc. in deutscher Sprache handhabt. Die meisten Wörter sind in der deutschen Sprache länger als in der englischen, so daß stellenweise Abkürzungen bei der Eingabe verwendet werden müssen. Hier die neuen Basic-Befehle:

```
END = AUS        FOR = FUER       NEXT = NAECHST      DATA = WERT
INPUT = EIN      READ = LES       LET = LASS          GOTO = GEH
RUN = LAUF       IF = WENN        RESTORE = ANFANG    TO = BIS
REM = BEM        STOP = HALT      RETURN = ZURUECK    ON = MIT
WAIT = WARTE     LOAD = LADE      SAVE = SPEICHER     OPEN = AUF
POKE = POK       PRINT = DRUCK    CONT = WEITER       CLR = LOE
```




CMD = KOM	SYS = RUF	VERIFY = PRUEF	CLOSE = ZU
GET = HOL	NEW = NEU	GOSUB = UNTER	SPC = LEE
THEN = DANN	NOT = NICHT	STEP = IN	AND = UND
OR = ODER	INT = GNZ	ABS = BTR	SQR = WRZ
RND =ZUF	LOG = LN	PEEK = PEK	LEN = LNG
STR$ = KET$	VAL = ZAH	CHR$ = ZCH$	LEFT$ = LI$

RIGHT$ = RE$ MID$ = MI$

Das Programm wird durch LOAD "GERMAN BASIC",8,1 geladen und durch RUN gestartet. Es belegt den Speicherraum zwischen $801 und $C20. Gewöhnen Sie sich an die neuen Befehle?

(C. Zwerschke/H. Ponnath/ef)

```
Name : german basic      0801 0c22
-----------------------------------
0801 : 1e 08 c1 07 9e 32 30 38  21
0809 : 30 14 14 14 14 14 14 14  25
0811 : 47 45 52 4d 41 4e 20 42  c5
0819 : 41 53 49 43 00 00 00 78  b0
0821 : a0 00 84 fb a9 a0 85 fc  11
0829 : b1 fb 91 fb c8 d0 f9 e6  84
0831 : fc a5 fc c9 c0 d0 f1 a9  26
0839 : e0 85 fc b1 fb 91 fb c8  1f
0841 : d0 f9 e6 fc d0 f5 b9 b7  7a
0849 : 08 99 9e a0 b9 b7 09 99  8a
0851 : 9e a1 b9 a3 0a 99 8a a2  7f
0859 : c8 d0 eb a9 69 8d 66 a4  9f
0861 : a9 77 8d 75 a4 8d ac e1  05
0869 : a9 72 8d c3 bd a9 49 8d  90
0871 : 70 e4 b9 a3 0b 99 be f0  90
0879 : c8 c0 68 d0 f5 a0 09 b9  d1
0881 : 0a 0c 99 e6 ec 88 d0 f7  1b
0889 : a9 13 8d bd f5 a9 4a 8d  c8
0891 : d3 f5 a9 59 8d d9 f5 a9  c7
0899 : 4f 8d 94 f6 a9 60 8d 51  29
08a1 : f7 a9 18 8d 1d f8 a9 2c  bd
08a9 : 8d 3e f8 a9 e5 85 01 8d  72
08b1 : d6 fd 58 6c 00 a0 41 55  de
08b9 : d3 46 55 45 d2 4e 41 45  dc
08c1 : 43 48 53 d4 57 45 52 d4  2a
08c9 : 45 49 4e a3 45 49 ce 44  1d
08d1 : 49 cd 4c 45 d3 4c 41 53  08
08d9 : d3 47 45 c8 4c 41 55 c6  6c
08e1 : 57 45 4e ce 41 4e 46 41  6a
08e9 : 4e c7 55 4e 54 45 d2 5a  a9
08f1 : 55 52 55 45 43 cb 42 45  93
08f9 : cd 48 41 4c d4 4d 49 d4  4b
0901 : 57 41 52 54 c5 4c 41 44  64
0909 : c5 53 50 45 49 43 48 45  8f
0911 : d2 50 52 55 45 c6 44 45  71
0919 : c6 50 4f cb 44 52 55 43  07
0921 : 4b a3 44 52 55 43 cb 57  e7
0929 : 45 49 54 45 d2 4c 49 53  2c
0931 : d4 4c 4f c5 4b 4f cd 52  c3
0939 : 55 c6 41 55 c6 5a d5 48  13
0941 : 4f cc 4e 45 d5 54 41 42  bc
0949 : a8 42 49 d3 46 ce 4c 45  76
0951 : 45 a8 44 41 4e ce 4e 49  4b
0959 : 43 48 d4 49 ce ab ad aa  75
0961 : af de 55 4e c4 4f 44 45  01
0969 : d2 bc bd be 53 47 ce 47  1a
0971 : 4e da 42 54 d2 55 53 d2  12
0979 : 46 52 c5 50 4f d3 57 52  f9
0981 : da 5a 55 c6 4c ce 45 58  b7
0989 : d0 43 4f d3 53 49 ce 54  ac
0991 : 41 ce 41 54 ce 50 45 cb  30
0999 : 4c 4e c7 4b 45 54 a4 5a  a6
09a1 : 41 c8 41 53 c3 5a 43 48  ae
09a9 : a4 4c 49 a4 52 45 a4 4d  d7
09b1 : 49 a4 47 45 c8 00 5a 55  67
09b9 : 20 56 49 45 4c 20 46 49  71
09c1 : 4c 45 d3 46 49 4c 45 20  ba
09c9 : 4f 46 46 45 ce 46 49 4c  52
09d1 : 45 20 5a d5 46 49 4c 45  e2
09d9 : 20 4e 49 43 48 54 20 47  11
09e1 : 45 46 55 4e 44 45 ce 4b  a9
09e9 : 45 49 4e 20 47 45 52 41  d5
09f1 : 45 d4 4b 45 49 4e 20 45  2e
09f9 : 49 4e 47 41 42 45 20 46  bf
0a01 : 49 4c c5 4b 45 49 4e 20  63
0a09 : 41 55 53 47 41 42 45 20  2e
0a11 : 46 49 4c c5 4b 45 49 4e  68
0a19 : 20 46 49 4c 45 4e 41 4d  9f
0a21 : c5 49 4c 4c 45 47 41 4c  54
0a29 : 45 20 47 45 52 41 45 54  e6
0a31 : 45 4e 55 4d 4d 45 d2 4e  83
0a39 : 41 45 43 48 53 54 20 4f  ee
0a41 : 48 4e 45 20 46 55 45 d2  d0
0a49 : 53 59 4e 54 41 d8 [illegible] 55  56
0a51 : 52 55 45 43 4b 20 4f 48  8b
0a59 : 4e 45 20 55 4e 54 45 d2  3f
0a61 : 45 4e 44 45 20 44 45 52  65
0a69 : 20 57 45 52 54 c5 4c 4c  0e
0a71 : 45 47 41 4c 45 20 47 52  4b
0a79 : 4f 45 53 53 c5 55 45 42  4b
0a81 : 45 52 4c 41 55 c6 53 50  a4
0a89 : 45 49 43 48 45 52 20 56  61
0a91 : 4f 4c cc 55 4e 44 45 46  8d
0a99 : 49 4e 2e 20 41 55 53 44  2e
0aa1 : 52 55 43 cb 4c 4c 45 47  b3
0aa9 : 41 4c 45 52 20 49 4e 44  ba
0ab1 : 45 d8 55 4d 44 49 4d 45  b0
0ab9 : 4e 53 49 4f 4e 2e 20 46  50
0ac1 : 45 4c c4 44 49 56 49 53  f9
0ac9 : 49 4f 4e 20 44 55 52 43  10
0ad1 : 48 20 4e 55 4c cc 4c 4c  5d
0ad9 : 45 47 41 4c 45 52 20 4d  9e
0ae1 : 4f 44 55 d3 4c 4c 45 47  ed
0ae9 : 41 4c 45 52 20 54 59 d0  98
0af1 : 4b 45 54 54 45 20 5a 55  e8
0af9 : 20 4c 41 4e c7 46 49 4c  c6
0b01 : 45 20 44 41 54 45 ce 46  c7
0b09 : 4f 52 4d 45 4c 20 5a 55  57
0b11 : 20 4c 41 4e c7 47 45 48  ce
0b19 : 54 20 4e 49 43 48 54 20  42
0b21 : 57 45 49 54 45 d2 55 4e  d5
0b29 : 44 45 46 49 4e 2e 20 46  2e
0b31 : 55 4e 4b 54 49 4f ce 50  f6
0b39 : 52 55 45 c6 4c 41 44 c5  cb
0b41 : 9e a1 ab a1 b5 a1 bc a1  6d
0b49 : cf a1 da a1 eb a1 fc a1  d7
0b51 : 09 a2 1f a2 30 a2 36 a2  fe
0b59 : 48 a2 56 a2 65 a2 6e a2  47
0b61 : 7b a2 8c a2 9a a2 ab a2  57
0b69 : be a2 cc a2 d8 a2 e5 a2  7f
0b71 : ef a2 fd a2 0e a3 1f a3  47
0b79 : 24 a3 85 a3 0d 4f 4b 0d  d7
0b81 : 00 20 20 46 45 48 4c 45  b5
0b89 : 52 00 20 49 4e 20 00 0d  0d
0b91 : 0a 46 45 52 54 49 47 2e  63
0b99 : 0d 0a 00 0d 0a 52 2f 53  e4
0ba1 : 00 a0 45 2f 41 20 46 45  e1
0ba9 : 48 4c 45 52 a3 0d 53 55  4e
0bb1 : 43 48 45 a0 4e 41 43 48  0a
0bb9 : a0 0d 44 52 55 45 43 4b  5e
0bc1 : 20 50 4c 41 59 20 41 4d  7b
0bc9 : 20 42 41 4e c4 44 52 55  87
0bd1 : 45 43 4b 20 52 45 43 4f  8a
0bd9 : 52 44 20 55 4e 44 20 50  28
0be1 : 4c 41 59 20 41 4d 20 42  ac
0be9 : 41 4e c4 0d 4c 41 44 c5  90
0bf1 : 0d 53 50 45 49 43 48 45  bf
0bf9 : 52 a0 0d 50 52 55 45 46  5a
0c01 : c5 0d 47 45 46 55 4e 44  98
0c09 : 45 4e 4c 41 c4 0d 4c 41  19
0c11 : 55 46 0d 00 00 00 00 00  cd
0c19 : 00 00 00 00 00 00 00 4c  b2
0c21 : 45 ce 2e cd 41 4e 4e 20  13
```

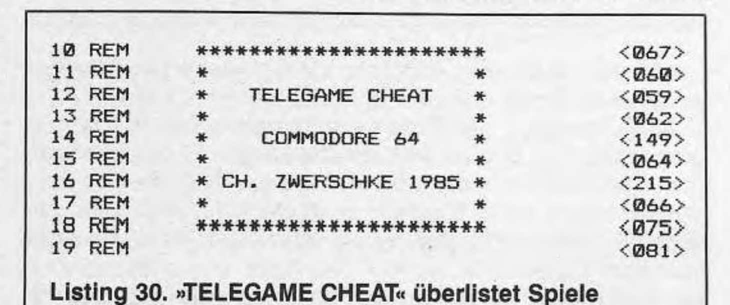
Listing 29. Nicht ganz erst gemeint ist »GERMAN BASIC«, das mit dem MSE einzugeben ist

21. Unverwundbar

»TELEGAME CHEAT« (Listing 30) ist ein kurzes Basic-Programm, das einen »Virus« in den C64 pflanzt, der ihn unempfindlich gegen Sprite-Kollisionen macht. Dadurch erhalten Sie bei Spielen mit einer Sprite-Spielfigur (Raumschiff, Pacman und so fort) die Unverwundbarkeit: Ohne Probleme können Sie jetzt durch alle Räumlichkeiten des Spiels laufen, fliegen oder schwimmen. Alle Widerwärtigkeiten wie Laserstrahlen, UFOs und Monster sind für Sie praktisch Luft geworden, und lange Spiele-POKE-Listen werden nahezu überflüssig.

TELEGAME CHEAT wird mittels RUN gestartet, dann wird das Spiel wie üblich geladen und gestartet. Keine Probleme hat TELEGAME CHEAT mit den »Oldies« unter den Spielen. Modernere Spiele könnten dadurch, daß sie superlang und auf komplizierte Weise komprimiert sind, eventuell Schwierigkeiten bereiten.

(C. Zwerschke/H. Ponnath/ef)



```
10 REM     **********************        <067>
11 REM     *                    *        <060>
12 REM     *   TELEGAME CHEAT   *        <059>
13 REM     *                    *        <062>
14 REM     *    COMMODORE 64    *        <149>
15 REM     *                    *        <064>
16 REM     * CH. ZWERSCHKE 1985 *        <215>
17 REM     *                    *        <066>
18 REM     **********************        <075>
19 REM                                   <081>
```

Listing 30. »TELEGAME CHEAT« überlistet Spiele

```
20 S=0:FOR I=679 TO 765:READ J                 <208>
30 S=S+J:POKE I,J:NEXT                         <087>
40 IF S<>13560 THEN PRINT"FEHLER!":END         <159>
50 POKE 680,PEEK(816):POKE 681,PEEK(817)       <021>
60 POKE 816,679 AND 255:POKE 817,679/256       <092>
70 PRINT:PRINT"TELEGAME CHEAT!"                <193>
100 DATA 32,165,244,176,81,132                 <131>
101 DATA 253,169,54,133,1,160                  <163>
102 DATA 0,132,251,169,8,133                   <046>
103 DATA 252,177,187,201,36,240                <070>
104 DATA 54,160,0,177,251,201                  <009>
105 DATA 173,208,32,200,177,251                <134>
106 DATA 201,30,240,4,201,31                   <130>
107 DATA 208,21,200,177,251,201                <230>
108 DATA 208,208,14,169,234,145                <150>
109 DATA 251,136,169,0,145,251                 <161>
110 DATA 136,169,169,145,251,230               <190>
111 DATA 251,208,2,230,252,165                 <124>
112 DATA 252,197,253,144,204,240               <173>
113 DATA 202,169,55,133,1,164                  <080>
114 DATA 253,24,96                             <015>
```

Listing 30. (Schluß)

22. LIST gestoppt

Die letzte veröffentlichte Routine zum Stoppen der Listing-Ausgabe, hatte eine Länge von 40 Byte und mußte mit »,8,1« geladen werden, worauf die Programmzeiger geändert wurden. Was ist aber zu tun, wenn eine Routine zum Anhalten des Listings nach dem Laden eines Programms gebraucht wird? Aus dieser Zwangslage entstanden folgende Basic-Zeilen, die im Direktmodus eingegeben werden:

```
0 DATA 174, 141 , 2, 202, 240, 250, 170, 76, 26,
  167, <RETURN>
FORA = 320 TO 329 : READ A: POKE I, A: NEXT:
POKE 774, 64: POKE 775, 1 <RETURN>
0 <RETURN>
```

Der LIST-Vektor in den Speicherstellen 774 und 775 wird auf die Adresse 320 gestellt, ab der ein Maschinensprache-programm von 10 Byte Länge liegt. Wenn beim Listen eines Programms die Taste <SHIFT> gedrückt wird, durchläuft das Programm eine Schleife und stoppt den LIST-Vorgang. Zum Abschalten von LIST-Hold sind folgende POKE-Befehle einzugeben:

```
POKE 774,26:POKE 775,167
```

Listing 31 zeigt den Quelltext der Routine.

(H. zur Nieden/ef)

```
-wait   ldx $028d ; shift-register
-       dex       ; =0, wenn shift
-       beq wait  ; warten, wenn shift
-       tax       ; status wiederherstellen
-       jmp $a71a ; zur list-routine
```

Listing 31. Klein, aber fein: LIST-HOLD © 64'er

23. Wie von Geisterhand...

...soll sich auch Ihr Computer künftig »selbst bedienen«, wenn er zu lange auf eine Eingabe warten mußte.

Angenommen, Sie haben ein interessantes Programm geschrieben (z. B. eine Textverarbeitung) und möchten zeigen, was dahinter steckt. So weit, so gut. Nun wird Ihr Programm kaum ohne Eingabe auskommen. Was aber geschieht, wenn der Computer auf etwas wartet und es nicht bekommt? Gar nichts, richtig. Das heißt, er wartet und wartet - und wartet. Und wenn Sie ihm »Auto-Input« (Listing 32) länger vorenthalten, wird er noch weiter warten müssen!

Die Funktionsweise von Auto-Input ist ebenso einfach wie genial: Um nach Ablauf einer vorgegebenen Zeit ein »Ausbrechen« aus der Eingabe in Basic zu ermöglichen, erfolgt diese über den Get-Befehl. Die Bedeutung der einzelnen Schritte ist in Tabelle 2 dokumentiert.

Zeile	Funktion
1000	Zurücksetzen von NA$ (geschieht vor jeder neuen Eingabe)
1010	Ausgabe des Cursors, Zurücksetzen der Uhr auf Null (geschieht bei jedem Tastendruck)
1020	Eingabewarteschleife: Ist die in TM vorgegebene Zeit abgelaufen, wird die Schleife verlassen
1030	Keine Eingabe (= Zeit abgelaufen) oder <RETURN> (= Eingabe abgeschlossen) führen zum Überspringen der Zeilen 1040 bis 1060
1040-1050	Vorbeugen von Fehlbedingungen: Korrekte Ausführung von <DELETE> und Lahmlegen der übrigen Steuerfunktionen (die ansonsten in NA$ übernommen und direkt ausgeführt würden; Beispiel <CRL/HOME>)
1060	Zulässige Zeichen werden in NA$ übernommen, anschließend Eingabe des nächsten Zeichens
1070	Löschen des Cursors. Ist die Eingabe noch leer, wird VO$ an NA$ übergeben
1080	Abschluß der Ausgabe und Rückkehr ins Hauptprogramm

Tabelle 2. So funktioniert Auto-Input

Vor dem Aufruf des Unterprogramms müssen den Variablen TM (maximale Wartezeit in Sekunden) und VO$ (Vorgabe für den Fall, daß keine Eingabe erfolgt) die entsprechenden Werte zugewiesen werden. Erfolgt nun innerhalb der Zeit TM keine Eingabe, wird der vorgebenene Wert VO$, ansonsten der eingebenene Wert an das betreffende Hauptprogramm in der Variablen NA$ zurückgegebenen. Bei der Eingabe von Zahlenwerten sind die Vorgaben im Stringformat (z.B. VO$= "123") an das Unterprogramm zu übergeben und nach der Rückkehr ins Hauptprogramm mit der VAL-Funktion ins Zahlenformat zu wandeln (siehe Zeile 340).

Listing 32 enthält neben der eigentlichen Routine, die bei Zeile 1000 beginnt, noch einige Zeilen, die Aufruf und Wirkung der Routine verdeutlichen sollen. Für die Eingaberoutine wurde ein stehender Cursor gewählt, um die Lauffähigkeit sowohl auf C 64 und C 128 als auch C 16 und Plus/4 zu erreichen

(Matthias Ullmann/ef)

```
100 REM * AUTO-INPUT * 11/88                   <024>
110 :                                          <086>
120 REM ---------------------------            <138>
130 REM  SIMULIERTES HAUPTPROGRAMM             <032>
140 REM ---------------------------            <158>
150 :                                          <126>
160 PRINT "{CLR}NAME{4SPACE}: ";               <041>
170 TM=1 : VO$="MATTHIAS ULLMANN"              <083>
180 GOSUB 1000                                 <136>
190 :                                          <166>
200 PRINT "{DOWN}STRASSE : ";                  <011>
210 TM=3 : VO$="HELMHOLTZSTR. 53"              <169>
220 GOSUB 1000                                 <176>
230 :                                          <206>
240 PRINT "{DOWN}PLZ/ORT : ";                  <047>
250 TM=5 : VO$="6200 WIESBADEN 1"              <077>
260 GOSUB 1000                                 <218>
270 :                                          <248>
280 PRINT "{DOWN}TELEFON : ";                  <216>
290 TM=3 : VO$="" : REM KEINE VORGABE          <037>
300 GOSUB 1000                                 <002>
310 :                                          <032>
320 PRINT "{DOWN}CODENR. : ";                  <119>
```

Listing 32. »Auto-Input« wartet nicht lange...

```
330 TM=1 : VO$="640815" : REM ZAHL          <139>
340 GOSUB 1000 : NA=VAL(NA$)                <229>
350 :                                       <072>
360 END                                     <108>
370 :                                       <092>
380 :                                       <102>
390 REM ---------------                     <010>
400 REM  UNTERPROGRAMM                      <019>
410 REM ---------------                     <030>
420 :                                       <142>
1000 NA$=""                                 <098>
1010 PRINT"{RVSON,SPACE,RVOFF}"; : TI$="00
     0000"                                  <181>
1020 GET A$ : IF A$="" THEN A$=CHR$(0) : I
     F TI/60<TM THEN 1020                   <233>
1030 A=ASC(A$) : IF A=0 OR A=13 THEN 1070   <104>
1040 IF A=20 AND NA$<>"" THEN NA$=LEFT$(NA
     $,LEN(NA$)-1) : PRINT A$;              <114>
1050 IF A<32 OR A>90 THEN PRINT"{LEFT}"; :
     GOTO 1010                              <052>
1060 NA$=NA$+A$ : PRINT"{LEFT}";A$; : GOTO
     1010                                   <009>
1070 PRINT CHR$(20); : IF NA$="" THEN NA$=
     VO$ : PRINT NA$;                       <068>
1080 PRINT : RETURN                         <095>
```

Listing 32. (Schluß)

24. Gelöschtes Programm zurückholen

Es passiert öfter, daß man versehentlich NEW eingibt oder einen Reset auslöst. Gibt es eine Möglichkeit, das gelöschte Programm zu retten? (Jörg-Dieter Richter)

Vorausgesetzt, daß Sie nach dem versehentlichen Löschen den Computer noch nicht abgeschaltet haben, keine weiteren Programmzeilen eingegeben und keine Variablen definiert haben, gibt es Hoffnung.

Damit Sie in einem solchen Fall das Programm zurückholen können, sollten Sie so bald wie möglich das unten abgedruckte Listing 33 mit dem MSE (siehe Seite 159) eingeben und auf einer Diskette speichern. Im »Ernstfall« nehmen Sie diese Diskette und geben ein:

```
LOAD "RENEW",8,8
NEW
SYS 828
```

Nach jedem dieser drei nacheinander einzugebenden Befehle ist die RETURN-Taste zu drücken. Jetzt steht das Programm wieder einsatzbereit im Speicher und sollte vor der weiteren Bearbeitung zunächst sicherheitshalber gespeichert werden. (Nikolaus Heusler/ef)

```
Name : renew                    033c 0379
-------------------------------------------
033c : a0 01 98 91 2b 20 33 a5   81
0344 : 18 a5 22 69 02 85 2d 85   f1
034c : 2f 85 31 a5 23 69 00 85   c7
0354 : 2e 85 30 85 32 20 1d a8   ec
035c : a2 19 86 16 20 87 a6 a2   0d
0364 : 09 bd 6f 03 20 d2 ff ca   b6
036c : 10 f7 60 0d 2e 4b 4f 20   ec
0374 : 57 45 4e 45 52 ff 00 00   cf
```

Listing 33. »RENEW« dient zum Zurückholen von gelöschten Programmen. Bitte dem MSE (Seite 159) eingeben.

25. GOTO X

Beim »strukturierten Programmieren« in Basic kann der Befehl GOTO X (Sprung zu einer variablen, berechneten Zeilennummer) gute Dienste leisten. Doch leider versteht der Basic-Interpreter des C64 nicht diesen nützlichen Befehl. Dieses nur 12 Byte lange Maschinenprogramm (Listing 34) erlaubt es, nun auch zu berechneten Zeilennummern zu springen. Im Gegensatz zu vielen anderen GOTO X-Routinen wird hierbei jedoch nicht das gesamte Basic-ROM kopiert (Listing 35). Daher die sparsame Nutzung des eh so knappen Speicherplatzes.

Aufgerufen wird das Programm mit »SYS 828, Zeilennummer«.

Verwendete Kernel-Routinen:

$AEFD prüft, ob ein Komma folgt.
$AD8A holt einen Ausdruck (hier die Zeilennummer).
$B7F7 wandelt den Ausdruck in Low- und High-Byte um.
$A8A3 ist die Adresse des GOTO-Befehls.

Ein kurzes Beispielprogramm:

```
10 INPUT "ZAHL VON 0 BIS 2 EINGEBEN",A
20 X=A+40: PRINT"SIE HABEN"
30 SYS 828,X
40 PRINT"0 GEDRUECKT":END
41 PRINT"1 GEDRUECKT":END
42 PRINT"2 GEDRUECKT":END
```

(Jörg Piller/ef)

```
Name : goto x     .obj     033c 0348
-------------------------------------
033c : 20 fd ae 20 8a ad 20 f7   91
0344 : b7 4c a3 a8 02 85 2d 85   2b
```

Listing 34. Berechneter GOTO-Sprung in nur 12 Byte Länge. Bitte Eingabehinweise auf Seite 159 beachten.

```
100 ;
110 ;  GOTO X
120 ; --------
130 ; AUFRUF : SYS 828,ZEILENNUMMER
140 ;
200 .BA 828
220 .EQ CKCOM =$AEFD
230 .EQ FRMNUM=$AD8A
240 .EQ GOTO  =$A8A3
250 .EQ CHANGE=$B7F7
260 ;
270  JSR CKCOM;KOMMA ?
280  JSR FRMNUM;ZEILE
290  JSR CHANGE;FORMAT
300  JMP GOTO;ZU GOTO
```

Listing 35. Dokumentierter Quelltext zu Listing 34

26. 59390 Bytes free

Mit Double-Basic (Listing 36) können Sie zwei unabhängige Basic-Programme im Speicher halten, die zusammen 59390 Byte (!) lang sein können. Die Programme können 38911 (PRG 1) und 20480 Byte (PRG 2) belegen.

Durch <CTRL> + <F1> werden beide Programme gegeneinander ausgetauscht, während alle Variablen des jeweiligen Programmes erhalten bleiben. Wenn Sie also erst Programm 1 bearbeitet haben, können Sie nach <CTRL> + <F1> mit Programm 2 fortfahren und umgekehrt. Zusätzlich besitzt Double-Basic zwei andere, ungewöhnliche Funktionen:

1. <CTRL> + <←> verlangsamt die Ausführungszeiten des Computers. Der Grad der Verzögerung kann durch »POKE 53024,...« eingestellt werden (255 = größte Verzögerung).

2. <CTRL> + <RUN/STOP> hält den Computer an, bis SPACE gedrückt wird. Sehr praktisch für das Listen eines Programmes: <CTRL> + STOP gedrückt halten und für jede nächste Zeile zusätzlich kurz SPACE drücken.

Wenn Sie Double-Basic mit dem MSE abgetippt haben, speichern Sie es. Nach dem Laden mit »,8,1« und NEW können Sie Double-Basic mit SYS 52992 initialisieren (SYS 52992 ist nach jedem <RUN/STOP RESTORE> notwendig!). Es stehen nun die beschriebenen Funktionen zur Verfügung.

Hinweise: <CTRL> + <F1> ist nur möglich, wenn sich der Computer im Direktmodus befindet. Während des Listens führt der Druck dieser Tastenkombination zum Absturz! Also: erst STOP, dann <CTRL> + <F1>. Da alle Variablen erhalten bleiben, kann ein Programm nach einem Tausch mit CONT fortgeführt werden, wenn es vorher unterbrochen wurde. Es sei denn, das System hat seit dem letzten Tausch eine FOR-, NEXT- oder GOSUB-RETURN-Struktur bearbeitet. In diesem Fall reagiert der Computer, wenn er auf NEXT oder RETURN stößt, mit Unsinn, Fehlermeldungen oder gar Absturz.

Einige Daten zu Double-Basic. Double-Basic ist in den System-Interrupt eingebunden und belegt den Speicherbereich von $CF00 bis $D000, ist also verwendbar mit Turbo-Tape und SMON (ohne Disk-Monitor).

In der Interrupt-Routine (ab $CF0B) wird auf die Betätigung einer der drei Tasten-Kombinationen geprüft. Wird der Programmtausch aufgerufen, erfolgt zuerst noch ein Test, ob sich der Computer auch im Direktmodus befindet. Daraufhin wird der Inhalt des Speichers von 2049 bis 22528 mit dem des RAM-Bereichs unter dem ROM vertauscht. Genauso wird mit den Basic-Vektoren in den Zellen 43 bis 66 (die unter anderem die Programmlänge und den freien Speicherplatz angeben) und einer Tabelle ab $CFD9 verfahren. Bevor in die normale Interrupt-Routine gesprungen wird, wird noch die Nummer des aktuellen Programmes angezeigt.

(Julian Ziersch/ef)

```
Name : dbasic                 cf00 d000
---------------------------------------
cf00 : a9 0b 8d 14 03 a9 cf 8d  ed
cf08 : 15 03 60 ad 8d 02 c9 04  85
cf10 : f0 03 4c 31 ea a5 cb c9  5a
cf18 : 04 f0 29 c9 39 d0 13 a0  bf
cf20 : 1e a2 ff ca ea ea ea ea  70
cf28 : ea ea ea d0 f6 88 d0 f1  37
cf30 : f0 e0 c9 3f d0 dc a9 7f  84
cf38 : 8d 00 dc a9 10 2d 01 dc  5a
cf40 : d0 fb f0 ce a5 9d c9 80  93
cf48 : d0 c8 ad fa cf c9 31 f0  39
cf50 : 08 a9 31 8d fa cf 4c 5e  47
cf58 : cf a9 32 8d fa cf 78 a9  9d
cf60 : 34 85 01 a9 01 85 fb a9  4c
cf68 : 08 85 fc a9 00 85 fd a9  1f
cf70 : a0 85 fe a2 20 20 88 cf  ab
cf78 : a0 00 84 fd a9 d0 85 fe  2e
cf80 : a2 30 20 88 cf 4c a6 cf  ed
cf88 : a0 00 b1 fb 85 02 b1 fd  3f
cf90 : 91 fb a5 02 91 fd e6 fb  65
cf98 : d0 02 e6 fc e6 fd d0 ea  3a
cfa0 : e6 fe ca d0 e5 60 a2 37  2c
cfa8 : 86 01 a2 18 b5 2b bc d9  b6
cfb0 : cf 9d d9 cf 94 2b ca 10  ac
cfb8 : f3 ac 21 d0 c8 a2 28 a9  f9
cfc0 : 20 9d ff 03 ca d0 fa a2  73
cfc8 : 0d bd f1 cf 9d 01 04 98  4d
cfd0 : 9d 01 d8 ca d0 f3 4c 31  bd
cfd8 : ea 01 08 03 08 03 08 03  64
cfe0 : 08 00 58 00 58 00 58 30  46
cfe8 : ff 00 2a 2a 2a 20 10 12  bf
cff0 : 07 00 2a 2a 2a 20 10 12  cf
cff8 : 07 2d 31 20 2a 2a 2a df  43
```

Listing 36. Double-Basic müssen Sie mit dem MSE eingeben

27. Z und Y vertauscht

Jeder, der einmal einen Schreibmaschinenkurs absolviert hat, hat sich auch schon über die Vertauschung der Tasten »Y« und »Z« auf der Tastatur des C 64 geärgert. Vor allem, wenn man in einem Basic-Programm in PRINT-Zeilen einen kleinen Text schreiben möchte, ist dieser Umstand sehr störend. Eine naheliegende, wenn auch sehr umständliche Methode ist, das Betriebssystem zu ändern. Es empfiehlt sich, diese Änderung dann in ein EPROM zu »brennen«.

Um die genannten Tasten in die deutsche Norm zu bringen, ist zuerst das Betriebssystem ins RAM zu kopieren (zum Beispiel mit einem Monitor), dann sind die folgenden POKEs einzugeben:

```
POKE 60301,89
POKE 60314,90
POKE 60366,217
POKE 60379,218
POKE 60431,183
POKE 60444,173
```

Übrigens: Wenn das Betriebssystem ins RAM kopiert wurde, besteht im allgemeinen das Problem, diese Kopie dauerhaft eingeschaltet zu lassen. Wenn man nämlich <RUN/STOP RESTORE> drückt, wird das RAM aus- und das ROM wieder eingeblendet. Durch einen POKE 64982,53 wird dies verhindert.

(Stefan Zellin/ef)

28. Gleichungen lösen

Eine verbesserte Form für ein Programm, mit dem Gleichungen gelöst werden können, stellen wir Ihnen mit Listing 37 vor.

Die linke und rechte Seite der Gleichung werden wie in Ausgabe 7/87 des 64'er-Magazins mit »DEF FN« definiert (Zeilen 10 und 20). Dann wird die Eingabe eines Startwertes für die folgende Iteration verlangt. Das Programm bildet die Differenz der beiden Funktionswerte und berechnet dann mit Hilfe der »REGULA FALSI« (Nullstellenalgorithmus) die Lösung der Gleichung. Einzelheiten finden Sie im Basic-Listing.

Durch Vorgabe des Startwertes ist es auch möglich, mehrere Lösungen der Gleichung zu finden.

(Markus Hagen/ef)

```
10 DEF FN A(X)=X*X*X-2*X*X+7          <098>
20 DEF FN B(X)=X*X+7*X                <043>
30 GE=5:REM GENAUIGKEIT               <187>
40 DE=0.5/10↑GE                       <025>
50 INPUT"STARTWERT";XA                <254>
60 X=XA                               <223>
70 FA=FN A(X)-FN B(X)                 <004>
80 X=XA+0.1                           <069>
90 F=FN A(X)-FN B(X)                  <191>
100 XN=X-(XA-X)*F/(FA-F)              <226>
110 D=ABS((X-XN)/(XN+DE))             <213>
120 IF D<DE THEN 140                  <008>
130 X=XN:GOTO 90                      <177>
140 PRINT"X=";XN                      <072>
```

Listing 37. Gleichungen lösen mit »gleichungen II«

29. 7.56 minus 1.56=?

Ein Leser schickte uns einen Brief mit einem interessanten Problem. Er hatte ein Programm geschrieben und war daran fast verzweifelt:

```
10 A=7.56:B=1.56:C=A-B
20 IF C=6 THEN PRINT "C IST 6!":END
30 PRINT "C IST UNGLEICH 6!"
```

Die große Preisfrage: Was ergibt 7.56 minus 1.56? 6? Nicht für den C 64! Wer's nicht glaubt, sollte das Programm ausprobieren. Aber damit nicht genug.

Fügen Sie folgende Zeile ein:

```
15 PRINT C
```

Laut dieser Zeile enthält die Variable C doch nur den Wert 6 ohne irgendwelche Nachkommastellen, die man von unserem »Rechengenie« ja inzwischen gewöhnt ist.

Nun gut, ein INT-Befehl müßte das Problem lösen:

```
17 C=INT(C)
```

Wahrscheinlich werden Sie in leises Schluchzen verfallen, wenn Sie das so geänderte Programm starten - C ist immer noch ungleich 6.

Scherz beiseite. (Thomas Röder/ef)

Hier ist die Lösung:

Jeder, der mit Fließkomma-Zahlen arbeitet, muß wissen, daß die interne Fließkomma-Darstellung in den wenigsten Computern exakt ist. Programmiersprachen (also auch Basic) machen Rundungsfehler, wenn es Fließkomma-Zahlen betrifft. Es wundert also nicht, daß (7.56-1.56) nicht gleich 6 ist, denn in jedem der folgenden Schritte werden Rundungsfehler gemacht (jedenfalls im 3. Schritt):

1. A = 7.56
2. B = 1.56
3. C = A-B

»PRINT C« liefert zwar 6, aber »PRINT C-6 liefert nicht 0, sondern -1.86264515 E -09 (ungefähr -0.00000000186264515).

C ist jetzt also ungefähr 5.99999999813735485 geworden. Indem man C = INT(C) hinzufügte, wird C also immer noch nicht gleich 6, sondern gleich 5, da die Nachkommastellen einfach abgeschnitten werden.

Es ist besser, C zu runden: C = INT(C+.5)

Folgendes Programm ergibt das erwünschte Resultat:

```
10 A=7.56:B=1.56:C=A-B
15 C=INT(C+.5)
20 IF C=6 THEN PRINT "C IST 6!":END
30 PRINT "C IST UNGLEICH 6"
```

(E. Polak/ef)

30. Wieviel Tage hat der Monat?

In vielen Anwendungen muß ein Tagesdatum (meist in der Form TTMMJJ - Tag Tag Monat Monat Jahr Jahr) eingegeben werden. Besonders dann, wenn mit dem Datum weitergerechnet werden soll, muß es »richtig« sein - der 30. Februar kann schon einige Berechnungen ins Nirwana leiten. Nun kann der C 64 nicht feststellen, ob das Datum inhaltlich richtig ist. Er kann aber durchaus prüfen, ob es sich bei dem eingegebenen um ein mögliches Datum handelt. Dabei muß zum Beispiel sichergestellt sein, daß sowohl Tag als auch Monat in den gültigen Intervallen liegen (1 bis Anzahl Tage des Monats und 1 bis 12), wobei ein eventuelles Schaltjahr noch zu berücksichtigen ist.

Eine programmtechnische Lösung liegt zum Beispiel darin, die Tagesanzahl pro Monat in einem Integer-Array [1...12] festzulegen und bei einem Schaltjahr über eine IF-Abfrage den Februar-Tageswert um 1 zu erhöhen.

Eleganter ist aber ein geschlossener Ausdruck für die Anzahl von Tagen (D) pro Monat als Funktion des Monats (M) und des Jahres (Y), wie er in dem folgenden Einzeiler wiedergegeben ist:

```
10 INPUT"JAHR, MONAT";Y,M:PRINT ((M -7 * INT
   ((M - 1)/ 7)) AND 1) + 30 + (M=2) * (2 + (Y / 4=
   INT(Y / 4)))
```

Der Basic-Ausdruck ist derart allgemein gehalten, daß er auf allen Commodore-Computern vom PET 2001 bis hin zum C128 laufen müßte. Auch eine Anpassung an GW-Basic und das Amiga-Basic ist einfach, wenn der »AND«-Operator durch Leerzeichen vom Rest der Zeile getrennt ist: »... /7)) AND 1) ...« und nicht, wie im Basic 2.0 bis Basic 7.0 erlaubt »... /7))AND1) ...«. (Dr. rer. nat. H. Haigis/ef)

31. Uhrzeit – Rechnung

Nach

```
DEF FN DZ(HR)=INT((INT(HR)+(HR-INT(HR))/.6)*10025.)
   100
```

ergibt FN DZ(17.30) den Dezimalwert 17,5, der für Berechnungen besser geeignet ist. Die umgekehrte Berechnung erledigt

```
DEF FN UR(DZ)=INT((INT(DZ)+(DZ-INT(Dz))*.6)*100+.5)/
   100
PRINT FN UR(17.25)
```

ergibt die normale Uhrzeit 17.15 Uhr, welche durch 17,25 dezimal dargestellt wird. (Alfred Poschmann/ef)

32. Zahlen raten

Dieses Programm zeichnet sich durch zwei Besonderheiten aus: seine Kürze und eine Erweiterung, die das Zahlenraten schwieriger und interessanter macht. Doch hier gleich die Spielregeln:

Der Computer denkt sich eine Zahl von 1 bis einschließlich 100. Der Spieler muß versuchen, diese Zahl zu erraten. Hierzu gibt der Spieler seinen Versuch ein. Der Computer gibt aus, ob der Versuch des Spielers größer oder kleiner als seine Zahl ist, beziehungsweise ob der Spieler die Zahl gefunden hat. Die ausgegebenen Zeichen haben folgende Bedeutungen: Ein »<« bedeutet, daß der Versuch des Spielers kleiner als die Zahl des Computers war, ein »>«, daß sie größer war und ein »=«, daß man die Zahl gefunden hat. Die Schwierigkeit ist, daß die zu erratende Zahl bei jedem Versuch um 3 erhöht wird. Ein Beispiel: Der Computer denkt sich die Zahl 48. Der Spieler gibt als ersten Versuch 50 an. Es kommt ein »>«. Beim nächsten Versuch ist jedoch nicht mehr die 48, sondern die 51 zu erraten (48+3). Die Angabe, daß die 50 größer als die zu erratende Zahl ist, stimmt jetzt also nicht mehr. Wenn die zu erratende Zahl größer als 100 geworden ist, so wird sie wieder auf 1 gesetzt.

Der Einzeiler lautet:

```
1x=int(rnd(1)*100)+1:fori=1to99:inputa:x=x+3+
   (x>97)*100:printchr$(61+sgn(a-x)):ifa<>x
   thennext
```

Der Einzeiler ist mit den Abkürzungen für die Basic-Befehle einzugeben:

```
1xint(rN(1)*100)+1:f01=1to99:inputa:x=x+3+(x>97)
*100:?cH(61+sG(a-x)):ifa<>xtHnE
```

Nach der Eingabe dieser Zeile ist mit dem Cursor zweimal nach oben zu gehen und erst dann die Return-Taste zu drücken, da ansonsten die Zeile nicht übernommen wird. (Michael Patra/ef)

33. <Control>-Tricks für den Wechsel der Zeichensätze

Sicherlich wissen die meisten C64-Fans, daß man durch Druck auf <SHIFT COMMODORE> zwischen Groß- und Kleinschrift (ohne SHIFT) hin- und herschalten kann. Innerhalb eines Basic-Programms läßt sich dieser Wechsel durch PRINT CHR$(14) und PRINT CHR$(142) erzwingen. Probiert es ruhig mal aus.

Was aber, wenn in einem Programm mit PRINT CHR$(8) der Wechsel zwischen beiden Zeichensätzen abgeschaltet wurde und man aber wechseln will? <SHIFT COMMODORE> funktioniert nun nicht mehr! Jetzt müßte man eigentlich das laufende Programm mit <RUN/STOP> oder <RUN/STOP RESTORE> unterbrechen und mit PRINT CHR$(9) eintippen. Es geht aber auch ohne Programmabbruch. Hierbei hilft die Control-Taste: Mit <CONTROL I> (<CTRL> und <I> gleichzeitig drücken) erlaubt man wieder den Wechsel der Zeichensätze, mit <CONTROL H> verbietet man ihn wieder. Wohlgemerkt, ohne das laufende Programm zu verlassen. Ist der Zeichensatz auf Groß/Grafik fixiert (<SHIFT COMMODORE> abgeschaltet), so kann man mit <CONTROL N> auf den Klein-/Großschrift-Zeichensatz wechseln. Ein Control-Code für den Wechsel zurück ist mir nicht bekannt, aber vielleicht findet ein Leser ja mehr heraus... Für Hinweise sind wir jederzeit dankbar.

(Alfred Poschmann/ef)

34. Diskette geschützt

Gegen einfaches Kopieren gibt es so gut wie keinen Schutz, da heutzutage (fast) alles kopiert werden kann. Wirkungsvoller ist aber ein Schutz gegen unberechtigten Zugriff auf eine Diskette. Die folgende Zeile formatiert eine Diskette, die danach gegen unberechtigten Zugriff geschützt ist (dabei werden alle Daten auf der Diskette gelöscht!):

```
OPEN 1,8,15,"M-W9"+CHR$(0)+CHR$(1):
PRINT #1,"N:name,id":CLOSE 1
```

Nun kann mit der leeren Diskette normal gearbeitet werden. Wird die Diskettenstation kurz aus- und eingeschaltet, ist der Schutz aktiv. Die geschützte Diskette verhält sich jetzt, als wäre sie unformatiert. Wollen Sie wieder auf Ihre Daten zugreifen, so ändern Sie die Speicherzelle für das Kennzeichen des Sektor-Headers:

```
OPEN 1,8,15,"M-W9"+CHR$(0)+CHR$(1):
CLOSE 1
```

Was bedeutet aber ein Schutz, dessen Auflösung jeder kennt?

Damit nur Sie Ihre Disketten lesen können, ist am Ende des OPEN-Befehls ein beliebiges Zeichen zu ergänzen (Ausnahme: CHR$(8)). Wer nun Ihre geschützten Disketten lesen möchte, hat im ungüstigsten Fall 255 Versuche!

Möchte man wieder auf eine normale (ungeschützte) Diskette zugreifen, ist die Diskettenstation kurz aus- und wieder einzuschalten.

(P. Schwendner/ef)

Spielen ohne Ende

Bevor Sie verzweifeln, weil Sie nie das Ende der Spiele aus den Sonderheften 37 oder 42 erreichen: In unserer kleinen Übersicht haben wir einige POKEs zusammengestellt, mit denen Sie fast jedes Spiel bewältigen.

In der Tabelle haben wir nur die Spiele aufgeführt, bei denen ein Trainer-POKE sinnvoll ist. Es fehlen also die Spiele, in denen beispielsweise zwei Spieler gegeneinander antreten. Bei Spielen, die in gepackter Version auf den Spieledisketten vorliegen, ist ein Trainer-POKE mitunter fast unmöglich. Beispiel: »Omega Force One«.

Die Eingabe der POKEs erfolgt entweder nach dem Laden des Programms im Direktmodus oder nach einem Abbruch des Spiels mit einem Reset. Hinweise finden Sie in der Übersicht.

ELON: POKE 2714,165 ; unendlich viele Leben

POKE 2063,0 ; nur ein Proton (funktioniert nur mit dem ersten POKE)

POKE 2070,234 ; immer nur ein Proton, keine Erhöhung

POKE 2699,0 ; Unsterblichkeit (Kollisionen sind ungefährlich)

Photon-Ranger: POKE 3200,173 ; kein Verlust der »Shields«

POKE 3182,212 ; Kollisionen sind ungefährlich

Weasel: POKE 2598,173 ; unendlich viele Leben

POKE 2571,0 ; Kollisionen sind ungefährlich

Mission X/II: Mit Reset aussteigen:

POKE 45904,255 ; 255 Leben

SYS 9712 ; neuer Spielstart

Battlefield: mit Reset aussteigen:

POKE 7301,0 ; unendlich viele Leben

SYS 2115 ; neuer Start

Omidar: POKE 15981,173 ; unendlich viele Leben

POKE 17771,0 ; unendlich viele Sprünge

Astromania: POKE 3126,255 ; 255 Leben

POKE 3886,173:POKE 6507,173 ; unendlich viele Leben (Diese POKEs werden nach dem ersten und vor dem zweiten RUN eingegeben)

Astropanic: Mit <RUN/STOP> <RESTORE> oder mit Reset aussteigen:

POKE 50309,173 ; unendlich viele Leben

SYS 49152 ; neuer Start

Sky Run: POKE 9667,169 ; unendlich viele Piloten

Hanoi+: POKE 4320,0:POKE 4322,0 ; Mogeln wird möglich

Blockbusters: In das Basic-Programm (File »Blockbusters«) sollte folgende Basic-Zeile eingebaut werden, bevor das Programm mit RUN gestartet wird:

1 POKE 16883,0

Damit stehen unendlich viele Bälle (Schläger) zur Verfügung.

Cookie-Eaters: POKE 2830,255 ; mehr Leben für jeden Spieler

Spiralon: POKE 5417,173 ; unendlich viele Leben (dauernde Anzeige »7«)

Waffles: POKE 6147,173 ; unendlich viele Aufklärer

POKE 3507,212 ; Kollisionen mit Gegner und Objekt werden harmlos

Iceball: POKE 5144,173 ; unendlich viele Bälle

(Nikolaus Heusler/ef)

Machen Sie mit!



Diesen Beitrag im 64´er-Sonderheft fand ich besonders gut:

Ausgabe: ________/________ Seite: __________

Artikel: ____________________________________

Ich wünsche mir für eine der folgenden Ausgaben folgende Themen:

Ich möchte an der redaktionellen Gestaltung mitarbeiten.

Meine Vorschläge:

Ich kann folgende(s) Programm(e) zur Veröffentlichung anbieten:

Dieses Problem habe ich:

Ich besitze einen: älteren C64 ___ neuen C64II ___

C128 ___ C128D (im Blechgehäuse) ___

mit Laufwerk(en): 1541(alt) ___ 1541c ___ 1541II ___

1570 ___ 1571 ___ 1581 ___

Ich verwende einen Drucker ___

mit 9 Nadeln ___ 24 Nadeln ___

Zusätzlich besitze ich einen

Amiga ___ PC ___ Atari ST ___ andere ___

Diese Note (1 bis 6, 1 am besten) gebe ich dem

64´er-Sonderheft: ___

Das sollte im 64´er-Sonderheft besser werden:

Name: ______________________________

Alter: ______________ Jahre

Adresse: ______________________________

Telefon: ______________________

Bitte schicken Sie die Mitmachkarte
in einem Briefumschlag
an folgende Adresse:

Markt & Technik Verlag AG
Redaktion Sonderhefte
Stichwort: Mitmachkarte 64´er
Hans-Pinsel-Straße 2
8013 Haar b. München

Schreiben Sie uns!



Crillion – der unendliche Spielespaß

Bild 1. Mit dem Programm »Crill-Edi« lassen sich auf einfache Weise viele interessante Level für »Crillion« zusammenstellen

Bild 2. Level 1 ist nach dem Einsatz des Editors erheblich schwieriger zu bewältigen als vorher

25 Level reichen für »Crillion-Fans« irgendwann nicht mehr aus. Ein Editor für weitere Level stellt Sie vor neue Herausforderungen.

Nur wer bei »Crillion« die Bewegung des Balles virtuos beherrscht, hat eine Chance, einen Highscore jenseits der Grenze von 100 000 Punkten zu erzielen. Ist das jedoch erst einmal geschafft, sind 25 Level zu wenig.

Mit dem Editor »Crill-Edi« (Listing 1) lassen sich die vorhandenen Level auf einfachste Weise verändern. Ihren Vorstellungen sind dabei keine Grenzen gesetzt.

Da das Programm gepackt und mit einem Autostart versehen ist, haben wir das Programm zum problemlosen Abtippen in einen anderen Speicherbereich verschoben. Haben Sie das Spiel mit dem MSE (Seite 159) eingegeben und auf einer Diskette gespeichert, kann es noch nicht sofort gestartet werden.

Um den Editor in ein startfähiges Programm zu wandeln, benötigen Sie das Programm »Change Crill-Edi« (Listing 3). Geben Sie das Programm mit Hilfe des Checksummers ein. Nach dem Laden mit

```
LOAD "CHANGE CRILL-EDI",8  <RETURN>
```

und Start des Programms werden Sie zunächst aufgefordert, die Diskette mit »Crill-Edi« ins Laufwerk einzulegen. Nach einem beliebigen Tastendruck wird zunächst die aktuelle Startadresse (dezimal 8192) auf dem Bildschirm angezeigt. Bei diesem Wert handelt es sich um die Startadresse des verschobenen Editors, der zum Abtippen in den Bereich ab $2000 verlegt wurde.

Anschließend wird diese Adresse automatisch auf den Originalwert 676 geändert. Sobald eine Meldung über die erfolgreiche Korrektur auf dem Bildschirm erscheint, läßt sich der Editor einwandfrei starten.

Kurzinfo: Crill-Edi

Programmart: Level-Editor für Crillion
Laden: LOAD "CRILL-EDI",8,1
Start: Das Programm startet automatisch.
Steuerung: Die Bedienung des Editors erfolgt über die Tastatur.
Besonderheiten: Nach dem Abtippen des Programms muß vor dem Start mit dem Zusatzprogramm »Change Crill-Edi« die Startadresse des Editors korrigiert werden. Beachten Sie bitte die entsprechenden Hinweise im Artikel.
Programmautor: Oliver Kirwa

Da die Crillion-Version, die auf der Spielediskette des Sonderhefts 37 enthalten ist, ebenfalls gepackt und mit Autostart versehen ist, läßt sie sich nicht editieren. Sie finden daher als Listing 2 eine ungepackte Version unter dem Namen »Crillion+«, die sich ohne Schwierigkeiten mit dem Editor bearbeiten läßt.

Laden Sie den Editor mit

LOAD "CRILL-EDI",8,1 <RETURN>

Nach einer kurzen Ladezeit, in der der Bildschirm abgeschaltet wird, meldet sich das Programm mit einem kleinen Auswahl-Menü (Bild 1).

Laden Sie zunächst mit <F3> das Spiel. Als Namen geben Sie »Crillion+« ein. Nach erfolgreichem Laden läßt sich mit <F1> der erste Level editieren (Bild 2).

Mit der Taste <**F3**> kommen Sie jeweils ein Level weiter, mit <**F5**> geht es zum vorherigen Level zurück. Kleiner Tip am Rande: Mit dieser Funktion lassen sich die vorhandenen Level in Ruhe studieren.

Mit <**F1**> wechselt der kleine Pfeil, der unter der Anzeige »Modus« dargestellt ist, seine Richtung. Die Pfeilrichtung bestimmt die Richtung, in der ohne Benutzung der Cursor-Tasten Steine gesetzt werden können. Diese Funktion erleichtert das Editieren über längere Strecken. Der Cursor, der aus einem kleinen Quadrat besteht, läßt sich wie gewohnt mit den Cursor-Tasten gezielt an jede Stelle des Spielfelds bewegen.

Mit der Taste <**F7**> verlassen Sie jederzeit den Editiermodus.

Die Farbe für ein zu setzendes Zeichen legen Sie mit <CTRL> und einer Zifferntaste von <3> bis <8> fest. Mit den Zifferntasten <1> bis <6> wählen Sie die verschiedenen Symbole aus, die an der entsprechenden Cursor-Position plaziert werden:

Taste	Symbol
1	Ein Leerzeichen
2	Block ohne Aufschrift
3	Mauer-Block
4	Totenkopf-Block
5	Block mit Stern
6	Diskette

Wenn Sie mit verschiedenen Farben arbeiten, nicht vergessen, zu jeder Farbe den entsprechenden Block mit Stern einzubauen. Nur beim Treffen des Balles auf einen solchen Block ändert der Ball seine Farbe.

Ist ein Level komplett, wird noch die Startposition des Balles festgelegt. Dies geschieht mit den Cursor-Tasten bei gleichzeitig gedrückter CBM-Taste (die Taste mit dem Commodore-Zeichen). Der Ball läßt sich mit den Cursor-Tasten über den ganzen Bildschirm bewegen.

Kurzinfo: Crillion+

Programmart: Geschicklichkeits-Spiel
Laden: Das Spiel wird vom Editor nachgeladen, läßt sich aber auch unabhängig mit »LOAD "CRILLION+",8,1« laden und mit RUN starten.
Steuerung: Der Ball wird mit einem Joystick in Port 1 gesteuert.
Programmautor: Oliver Kirwa

Kurzinfo: Change Crill-Edi

Programmart: Hilfsprogramm
Laden: LOAD "CHANGE CRILL-EDI",8
Start: Nach dem Laden RUN eingeben
Besonderheiten: Das Programm ändert automatisch die Startadresse von »Crill-Edi« von dezimal 8192 auf 676. Starten Sie das Programm, wenn Sie den Editor abgetippt und auf Diskette gespeichert haben. Nach der Korrektur wird das Hilfsprogramm nicht mehr benötigt.

Neue Level lassen sich mit einem kleinen Trick problemlos testen. Haben Sie mit <F7> den Editier-Modus verlassen, können Sie mit einem erneuten Druck auf <F7> das Spiel mit den neuen Leveln testen.

Den Start-Level (01, 05, 09, 13, 17 oder 21) bestimmen Sie mit dem Joystick. Haben Sie beispielsweise Level 15 editiert und wollen testen, ob es überhaupt zu schaffen ist, sollten Sie wie folgt vorgehen:

In Level 13 und 14 löschen Sie alle Symbole bis auf einen Block, den Sie noch treffen müssen, um in den nächsten Level zu gelangen. Auf diesem Weg gelangen Sie schnell in Level 15 und können es ausführlich testen. Sind Sie mit dem Level zufrieden, editieren Sie anschließend Level 14. Mit dem gleichen Vorgehen für die anderen Level stellen Sie fest, ob ein editiertes Level noch Fehler hat.

Was nützt ein Testspiel, wenn man bei einem festgestellten Fehler nicht sofort in den Editor zurückkehren kann? Kein Problem bei »Crill-Edi«: Ein Druck auf die RESTORE-Taste unterbricht das Spiel, auf dem Bildschirm erscheint das Auswahl-Menü des Editors. Unlösbare Level lassen sich damit schnell korrigieren.

Sind Sie mit allen neuen Level einverstanden, läßt sich das Spiel vom Menü aus mit <F3> auf Diskette speichern. Unter neuen Namen läßt sich eine interessante Sammlung von Crillion-Versionen zusammenstellen, mit denen »Crillion« nie langweilig werden kann.

Crillion-Fans kommen mit dem Editor voll auf ihre Kosten. Mit wechselnden Leveln können Sie sich immer neuen und anspruchsvolleren Herausforderungen stellen.

(ef)

```
Name : crill-edi          2000 2970
-----------------------------------
2000 : 20 15 fd 58 a9 9b 8d 00  e3
2008 : dd 8d 11 d0 20 aa f5 86  46
2010 : 2d 84 2e 20 53 e4 e8 20  4f
2018 : 71 a8 4c ae a7 2c 00 dd  5e
2020 : 30 fb 70 dc ad 00 dd 4a  ec
2028 : 4a ea ea 0d 00 dd 4a 4a  f0
2030 : ea ea 0d 00 dd 4a 4a ea  02
2038 : ea 0d 00 dd 60 4d 2d 45  14
2040 : 05 02 a9 8a 85 30 6c 30  ee
2048 : 00 a9 2b 8d 11 d0 ea a9  30
2050 : 08 20 b1 ff a9 6f 20 93  92
2058 : ff a0 f4 b9 ed 01 20 a8  d4
2060 : ff c8 d0 f7 20 ae ff 78  5f
2068 : 8c 00 dd a5 2b ea 85 ae  9e
2070 : 31 ea a5 2c ea 85 af 20  df
2078 : c1 02 91 ae e6 ae d0 f7  8b
2080 : e6 af d0 f3 ed 02 20 18  90
2088 : c1 a9 12 8d 05 1c a0 01  0a
2090 : b1 2e aa 88 b1 2e a0 04  2b
2098 : 85 0c 86 0d a9 80 85 03  21
20a0 : 58 a5 03 30 fc 78 ad 00  dc
20a8 : 06 48 d0 05 ad 01 06 85  ad
20b0 : 6c b9 00 06 49 ff aa 0a  0d
20b8 : 0a 0a 29 18 48 a9 18 8d  62
20c0 : 00 18 68 8d 00 18 8a 0a  97
20c8 : 29 18 ea 8d 00 18 8a 4a  e9
20d0 : aa 29 18 8d 00 18 8a 4a  46
20d8 : 4a 29 18 8d 00 18 c4 6c  1b
20e0 : 08 c8 a9 00 8d 00 18 28  40
20e8 : d0 c7 a0 02 ae 01 06 68  e0
20f0 : d0 a6 a9 10 8d 00 18 58  6a
20f8 : 4c 22 eb a9 01 08 0c 08  16
2100 : c3 07 9e 32 30 36 32 ff  b2
2108 : 00 00 00 78 a0 c5 b9 46  c3
2110 : 08 99 fe 00 88 d0 f7 84  9c
2118 : 01 84 ac 84 ad a2 04 b5  82
2120 : aa d0 02 d6 ab d6 aa ca  3f
2128 : ca d0 f4 b1 ae 91 ac a9  4b
2130 : 0c c5 ae a9 09 e5 af 90  9f
2138 : e4 a9 2c 85 ae a9 33 85  bd
2140 : af 4c ff 00 a2 de b1 ac  56
2148 : 20 b4 01 9d 32 01 e8 d0  27
2150 : f5 a9 08 85 60 a9 ea 85  d7
2158 : 5f a2 03 20 12 02 f0 29  15
2160 : c9 07 d0 15 20 10 02 d0  b0
2168 : 0b a2 04 20 12 02 69 07  ae
```

```
2170 : 85 5d 90 05 a2 0a 20 12  88
2178 : 02 20 b2 01 f0 71 20 bb  ea
2180 : 01 c6 5d d0 f4 c6 5e 10  75
2188 : f0 20 10 02 d0 27 a9 02  be
2190 : 85 61 a2 08 20 12 02 38  7b
2198 : a5 ae e5 5d 85 5d a5 af  f3
21a0 : e5 5e 85 5e b1 5d e6 5d  3e
21a8 : d0 02 e6 5e 20 bb 01 c6  70
21b0 : 61 d0 f1 f0 a4 20 10 02  a3
21b8 : d0 1a a9 03 85 61 20 10  64
21c0 : 02 d0 cf a2 0a 20 12 02  60
21c8 : 69 00 85 5d a5 5e 69 01  33
21d0 : 85 5e 90 c3 e8 20 12 02  fd
21d8 : 4a d0 04 69 04 d0 dd b0  58
21e0 : 07 20 12 02 69 06 d0 d4  70
21e8 : a2 08 20 12 02 90 cd a9  08
21f0 : 37 85 01 58 4c 2c 33 b3  90
21f8 : ac e6 ac d0 02 e6 ad 60  2b
2200 : 91 ae e6 ae d0 02 e6 af  90
2208 : 60 a2 01 86 5c 84 5d 84  33
2210 : 5e c6 60 d0 09 a9 08 85  0d
2218 : 60 20 b2 01 85 5f 06 5f  7f
2220 : 26 5d 26 5e c6 5c d0 e9  b1
2228 : a7 5d 60 03 a9 00 8d 1d  01
2230 : 3d a9 22 85 fe a9 71 85  89
2238 : fd a9 01 8d 16 78 41 80  27
2240 : 8d 8a 02 a9 aa 8d 13 5c  e4
2248 : 74 0f 8d 98 d4 20 9a e5  3f
2250 : 20 b8 33 9f ba 21 d0 8d  a0
2258 : 20 d0 20 30 35 a2 00 bd  d2
2260 : cb 3a 20 d2 ff e8 e0 e2  3b
2268 : d0 f5 91 91 14 47 84 bb  ce
2270 : 25 8d 17 3d a2 10 bd fb  63
2278 : 3c 9d 1f c1 ca d0 f7 20  d6
2280 : e4 ff c9 85 d0 06 20 03  4b
2288 : 34 20 3b 37 c9 86 d0 03  9c
2290 : 4c 50 34 c9 87 11 40 4c  e5
2298 : 75 a7 88 d0 de 91 be a3  da
22a0 : 56 35 a9 39 8d 7d 1a a9  a3
22a8 : a2 8d 82 1a 4c 16 0e 4c  3b
22b0 : 80 2f 45 07 8d 86 02 20  4f
22b8 : 44 e5 a2 14 a0 00 18 20  c5
22c0 : f0 ff 6a 3c 2e 5a 8d 9d  f9
22c8 : 6a c2 7b 8d ff 06 8d 15  b4
22d0 : 3c fc 06 a9 42 8d fd 0b  e0
22d8 : 20 fe 32 0d 0e 48 1f 07  53
22e0 : 1a 07 60 20 f9 33 78 e6  83
22e8 : fa 60 48 a9 ff 38 e9 01  c5
22f0 : d0 fb 68 60 ad 6a c5 d0  cb
22f8 : 2b a2 18 bb 9d 80 d4 0d  3d
2300 : 1d f8 e5 22 a9 f9 f8 02  a9
2308 : b0 21 8d 84 2f 10 14 8d  1c
2310 : 81 d4 a2 64 20 f2 33 28  e7
2318 : 50 f1 5e dd 60 a9 13 69  27
2320 : 40 a9 90 50 76 e1 08 a0  3b
2328 : 01 20 ba ff ad 12 69 e2  bf
2330 : 4e a0 3d 20 bd ff f1 1f  04
2338 : 55 f0 51 10 8d 12 35 20  da
2340 : c4 34 61 73 e9 5b 33 20  6c
2348 : 34 34 73 82 20 d5 ff b0  d6
2350 : 05 a9 01 08 22 89 0a 4a  96
2358 : 33 a6 45 12 03 01 15 78  ef
2360 : 3c 9d e7 04 5c 15 09 c0  f9
2368 : a1 ab 23 3c 78 a6 55 e2  07
2370 : f0 78 2d 8d 78 05 8d 79  72
2378 : 05 75 28 2a 0a a0 25 92  e7
2380 : 08 85 fc f8 28 85 fb a9  9b
2388 : fb 20 d8 ff b8 b0 c8 00  fe
2390 : 8c 46 38 00 eb 86 cc 02  78
2398 : 9f 36 27 90 07 c9 5b b0  bc
23a0 : 03 20 0e 35 c9 20 d0 c1  42
23a8 : cd 0d 42 20 01 78 14 d0  56
23b0 : 0b ae 2b 60 f0 06 ce 50  04
23b8 : 69 03 d2 d3 51 23 01 21  46
23c0 : 44 b5 a9 20 9d 40 05 60  fe
23c8 : 09 a0 e0 10 b0 09 ee 75  55
23d0 : 9d 4e 48 03 3c 70 31 16  3f
23d8 : ad 3b e5 4a 76 57 60 60  4a
23e0 : a2 12 20 ff e9 40 14 5f  43
23e8 : 80 c1 3c dd 05 08 d0 12  0c
23f0 : 33 d2 05 d0 f3 a9 35 8d  64
23f8 : ed 08 a9 55 8d ec 08 4c  f7
2400 : 0e 08 37 2d 39 ee 10 72  b6
2408 : 8d 01 1c a9 73 8d 0f 2b  88
2410 : 74 8d 1b 02 b0 75 8d 27  8d
2418 : 2f 39 17 8d a6 11 a9 1f  33
2420 : 8d 4b 0e a9 64 8d 4a 0e  04
2428 : d1 01 8d 49 64 a9 8d 9c  09
2430 : 0e 4c d0 9d 0e 25 64 8d  03
2438 : 9e 51 51 f4 11 e2 f5 05  7c
2440 : 60 1c 8d f6 53 c1 0f 25  ba
2448 : a9 fe 0f cc 60 a0 08 ae  56
2450 : 13 3d 7a e9 fd 88 d0 f7  35
2458 : 60 ad 00 d0 18 69 09 8d  b5
2460 : 0e d0 ad 01 c8 48 32 90  1b
2468 : d0 9b ef 06 0a a0 28 38  79
2470 : ed 11 c5 18 6d ee 06 8d  dc
2478 : 34 03 a8 60 08 a0 29 62  53
2480 : a9 3e 8d 57 20 b6 35 0f  41
2488 : 94 35 20 55 66 20 c9 35  63
2490 : e4 91 fd 1c 84 40 a0 a6  5a
2498 : 91 d4 a9 38 05 15 38 d0  80
24a0 : 4a 11 10 f2 8d 39 05 21  ce
24a8 : cc 35 12 03 8d 9e 20 57  f1
24b0 : 1a 5b 38 48 c9 e1 90 18  ad
24b8 : a9 fc 8d 15 eb 89 36 8a  de
24c0 : a1 ff 06 8c aa 30 4c 57  80
24c8 : 36 e4 b0 12 ee 00 a1 a0  95
24d0 : ba a9 aa 35 63 29 0f c9  ff
24d8 : 02 9c ee 9b 71 d2 d0 02  54
24e0 : f0 17 c9 16 b0 13 a9 f0  bd
24e8 : 21 c4 fd 64 a3 36 e8 2b  5d
24f0 : 4f ce 89 4c 41 e0 19 cd  ad
24f8 : d2 4b 3c 01 c6 80 40 64  d9
2500 : 68 f3 0e 1a 9e 46 90 12  ab
2508 : ce 25 1a 77 8e 15 3d 54  0e
2510 : 81 ae 9c bd e3 3c 1b 8e  71
2518 : e8 41 cf 0f 10 73 1b c4  09
2520 : 1c ec 27 1e 8a 0d ed 06  15
2528 : 8d 2c cd 00 fd 03 e0 09  cc
2530 : 0f 2e 6d 20 24 37 4c 13  09
2538 : 3a ad 16 3d c9 1a 90 05  30
2540 : 59 42 86 10 d0 51 19 c5  e5
2548 : a8 65 c7 bb 46 21 85 fe  8e
2550 : ae 16 4a 6e 1a 1d 61 03  80
2558 : c1 17 3d 8d db cd c9 25  43
2560 : 14 2e 20 22 36 4c 89 36  30
2568 : a5 cb c9 40 b8 c1 ca 26  7e
2570 : 01 c3 92 d4 8d 8f d4 a2  80
2578 : 00 8e 0d 93 ea 0d 94 8d  fa
2580 : 4a 03 30 49 d0 12 9e 58  4a
2588 : a9 93 89 ce 47 0c 13 4c  f1
2590 : 83 90 e0 6b e4 30 0f e4  d7
2598 : 00 f1 83 90 e0 86 e4 30  ba
25a0 : 37 e4 50 4c 1c f6 0b 19  bf
25a8 : cf 8d 1a cf 4e 37 04 d0  0f
25b0 : a2 07 a0 21 2b 4c f0 c6  88
25b8 : 83 8c 1d 83 8a d8 6c c5  e6
25c0 : d0 00 87 11 cc 40 aa 96  3b
25c8 : 7f cb 2a ae 40 e7 8d 96  34
25d0 : cb 8d 81 cb a9 81 8d 87  27
25d8 : 41 84 01 90 93 19 90 07  00
25e0 : 32 2e 8d f8 62 ff cf 9b  48
25e8 : 51 fa 04 71 81 fe cf a0  76
25f0 : a7 b1 fd 8d 29 d0 c8 81  e0
25f8 : 70 6c 19 71 05 d0 d1 73  18
2600 : 20 dd 35 46 55 27 d0 d6  a4
2608 : 00 2d d0 8d 2e c8 1c 2c  77
2610 : a2 c0 0d d0 4f 20 17 d2  68
2618 : 1d 26 22 14 6d ed 06 a0  f3
2620 : 23 a2 08 98 e1 00 f1 2c  e8
2628 : 64 cd bd fd e6 ae 8d 02  c0
2630 : 85 02 8a 29 02 d0 2a a5  19
2638 : 02 c9 03 70 c1 10 e0 00  0e
2640 : f0 06 20 58 36 45 64 20  a6
2648 : 22 1a 6c c9 07 d0 53 50  b0
2650 : ec a3 50 e4 89 18 6c a9  1d
2658 : 20 1b ad 78 81 e0 03 d0  4d
2660 : 0a c9 15 90 2f ce 7c 3b  78
2668 : 4c a4 38 c9 fa b0 25 ee  f5
2670 : 0a 14 d5 f1 1e ad 05 1f  d9
2678 : bb 3b 90 10 ce 05 15 80  61
2680 : e0 b0 06 ee a1 0f 41 95  da
2688 : 7e 3a 28 d0 15 ad 13 3d  cd
2690 : c9 5a b0 03 1c 41 ce 4a  ae
2698 : a2 04 72 a8 4c 31 4c 76  5a
26a0 : 2b 32 09 87 90 23 27 ee  b4
26a8 : c2 e3 2d 33 d0 42 12 9d  b0
26b0 : d0 06 23 28 0c a2 f0 8e  08
26b8 : 19 84 a0 de 8c 01 d0 ac  85
26c0 : 7c 00 8c 30 3d a0 01 8c  5b
26c8 : 13 5d a5 84 02 20 eb 35  bf
26d0 : 6f f9 95 a6 cb e0 33 f0  e8
26d8 : 04 e0 40 d0 09 c6 02 a4  8f
26e0 : 86 ea 9a 91 ac 28 a4 6a  28
26e8 : e0 04 8b f2 83 06 d0 31  1a
26f0 : a0 00 d9 0c 3d f0 08 c8  95
26f8 : c0 38 02 f6 4c 5b 39 c8  4a
2700 : c8 98 c1 5a 8a 8d 29 8e  a7
2708 : a7 91 fd b0 08 25 38 69  6b
2710 : 08 8d b5 5c 14 3d 20 34  ec
2718 : 37 d8 91 3c cc c9 38 d0  45
2720 : 0e ac 02 2e 01 2a 38 e3  d5
2728 : 86 f7 15 d0 05 a2 00 4c  07
2730 : bd 36 c9 08 09 39 05 09  fc
2738 : 34 0b 24 e4 0a 24 d0 10  bd
2740 : 93 0f 90 93 13 42 4e 14  96
2748 : 42 74 03 d0 32 20 2d 18  a8
2750 : c0 54 19 03 52 94 18 03  11
2758 : 5a 65 15 d0 a2 ff e0 e2  38
2760 : a9 3f 8d 02 dd a9 97 8d  f1
2768 : 00 02 e0 04 8d 88 02 a9  9a
2770 : 15 f0 86 57 15 85 c6 4c  bb
2778 : 3e 33 c9 05 d0 0c 36 31  0b
2780 : ee c1 e0 fc 36 4c 3b 37  48
2788 : c9 06 d8 81 ce ee 20 e2  65
2790 : 04 d0 11 06 a2 49 01 0a  8e
2798 : b9 8d 17 d7 09 42 8c 18  de
27a0 : c2 d6 a8 a0 a8 54 06 98  82
27a8 : b9 b7 0c d0 c8 ad 05 88  79
27b0 : 5a 0d d0 82 8f 26 60 92  e6
27b8 : 99 d5 c9 b0 c9 ae c2 c2  67
27c0 : ae 42 08 20 14 50 71 41  a2
27c8 : 0d db 14 34 ae 1c 73 16  14
27d0 : 19 12 74 9f c2 20 08 71  34
27d8 : 01 1d 42 d5 ca c9 0b 19  0c
27e0 : 0f 1c c5 80 43 94 ab cb  9e
27e8 : c2 9a ca cb b1 20 b1 ca  9c
27f0 : ca b1 10 c0 bd bd 20 a3  40
27f8 : 20 e0 08 01 e3 61 63 88  92
2800 : b0 e2 17 e0 d0 0d 57 52  7b
2808 : 49 54 54 45 4e 20 42 59  db
2810 : 20 4f 4c 49 56 45 52 20  2d
2818 : 4b 49 52 57 41 20 49 4f  60
```

Listing 1. Die Startadresse von »Crill-Edi« muß vor dem Start korrigiert werden. Geben Sie das Listing bitte mit dem MSE **(Seite 159)** **ein.**

```
2820 : 8f 31 39 38 38 08 9a 13   f2
2828 : 11 11 1d c3 20 38 20 20   06
2830 : 4d 20 45 20 3c 40 55 c7   8d
2838 : 9d b8 0a 11 0d 13 1e 20   f8
2840 : 20 3c 19 12 20 46 31 20   40
2848 : 92 91 16 34 a4 04 71 17   0d
2850 : 45 4c 45 a2 4c 31 26 44   d0
2858 : c2 49 62 06 e1 e1 b2 e3   d8
2860 : 33 e2 32 e2 43 94 28 4c   ff
2868 : c4 4f 4e 37 5e 41 44 1f   8e
2870 : 2d 3e 35 35 2d c5 53 50   9f
2878 : 45 49 43 48 f1 e1 d3 13   df
2880 : 37 e3 93 12 49 95 06 00   29
2888 : e1 14 04 82 9a 4b 47 58   97
2890 : 28 15 53 4b fa 03 6a 4c   8b
2898 : 4f 50 50 59 e3 c3 4e 44   6d
28a0 : 20 47 49 42 31 61 83 c9   be
28a8 : 58 41 4d c2 44 60 1d d0   aa
28b0 : 4c 2d 0b 8d 53 08 0d 80   b2
28b8 : 33 23 2c b0 3f 04 21 bb   ae
28c0 : 11 96 52 57 1b 00 a1 58   85
28c8 : c8 4f a8 80 2e 48 0d 04   d3
28d0 : f8 2d 7c 04 04 90 46 41   5f
28d8 : 71 53 53 20 81 e2 15 10   6f
28e0 : 66 74 55 45 42 28 33 40   31
28e8 : 60 02 41 18 78 65 37 96   59
28f0 : 2c 4d 55 53 53 0d 40 3a   96
28f8 : 20 56 4f 52 cb 9c 4d cb   d0
2900 : 8c 8d b0 54 45 48 3f 35   07
2908 : 21 13 10 05 09 03 08 05   2a
2910 : 12 14 fc 3f 00 03 5e c0   66
2918 : 03 15 03 1e 17 01 1e 1b   53
2920 : 18 1c 4a 87 88 40 76 9e   6b
2928 : 32 30 36 32 9f 8b 65 5d   ed
2930 : 11 c8 e3 87 11 4d 4f 44   d0
2938 : d5 5d 1e b1 0e 10 25 64   38
2940 : 65 66 67 80 71 70 17 55   64
2948 : 10 6c 6d 6e 6f 20 68 69   24
2950 : 6a 6b 40 72 73 74 75 f0   61
2958 : 18 01 ee ff 7e 3c 18 08   e7
2960 : 0c 0e ff ff 0e 0c 08 08   e5
2968 : 0b 10 13 18 1b 00 aa 00   a0
```

Listing 1. (Schluß)

```
Name : crillion+          0801 330e
-----------------------------------
0801 : 0d 08 00 00 20 9e 33 36   43
0809 : 30 36 21 00 00 00 00 bd   18
0811 : bd bd bd bd bd bd bd aa   ea
0819 : aa ff ff ff ff 55 55 aa   18
0821 : a9 fd fd fd fd 7d bd aa   20
0829 : aa bf bf bf bf bd bd bd   f7
0831 : bd fd fd fd fd 55 55 bd   87
0839 : be bf bf bf bf 95 55 7c   b6
0841 : c6 c6 de c6 c6 c6 00 fc   97
0849 : c6 c6 dc c6 c6 fc 00 7e   d3
0851 : c0 c0 c0 c0 c0 7e 00 fc   b3
0859 : c6 c6 c6 c6 c6 fc 00 fe   5f
0861 : 60 30 f8 c0 c0 fe 00 7e   30
0869 : c0 c0 dc c0 c0 c0 00 7e   e7
0871 : c0 ce c6 c6 c6 7c 00 c6   01
0879 : c6 c6 f6 c6 c6 c6 00 7e   d8
0881 : 18 18 18 18 18 7e 00 06   30
0889 : 06 06 06 06 c6 7c 00 c6   b3
0891 : cc d8 fc c6 c6 c6 00 c0   05
0899 : c0 c0 c0 c0 c0 7e 00 6c   da
08a1 : fe d6 c6 c6 c6 c6 00 fc   31
08a9 : c6 c6 c6 c6 c6 c6 00 7c   f8
08b1 : c6 c6 c6 c6 c6 7c 00 fc   af
08b9 : 06 06 fc c0 c0 c0 00 7c   24
08c1 : c6 c6 c6 de cc 76 00 fc   f2
08c9 : 06 06 fc d8 cc c6 00 7e   2c
08d1 : c0 c0 7c 06 06 fc 00 ff   19
08d9 : 00 18 18 18 18 18 00 c6   be
08e1 : c6 c6 c6 c6 c6 7c 00 c6   73
08e9 : c6 c6 c6 cc d8 f0 00 c6   00
08f1 : c6 c6 d6 d6 fe 6c 00 c6   8c
08f9 : c6 6c 38 6c c6 c6 00 c6   c1
0901 : c6 c6 76 06 06 fc 00 fe   cf
0909 : 0c 18 30 60 c0 fe 00 00   3d
0911 : 00 00 00 20 70 70 20 7e   1d
0919 : c3 9d b1 b1 9d c3 7e 00   3f
0921 : 60 00 00 00 60 00 00 00   88
0929 : 00 00 00 00 18 18 30 00   2d
0931 : 7c c6 ce de f6 e6 c6 7c   5a
0939 : 18 38 18 18 18 18 18 3c   92
0941 : fc 06 06 7c c0 c0 c0 fe   64
0949 : fe 0c 18 3c 06 06 06 fc   7e
0951 : c6 c6 c6 c6 7e 06 06 06   41
0959 : fe c0 c0 fc 06 06 06 fc   2a
0961 : 7e c0 c0 fc c6 c6 c6 7c   c6
0969 : fe 06 0c 18 18 18 18 18   43
0971 : 7c c6 c6 7c c6 c6 c6 7c   48
0979 : 7c c6 c6 c6 7e 06 06 fc   0d
0981 : 9f 93 11 11 11 11 11 11   50
0989 : 11 9d 9d 9d 9d 9d 9d 9d   fc
0991 : 9d 20 1e 53 43 4f 52 45   b3
0999 : 11 9d 9d 9d 9d 9f 30 30   8c
09a1 : 30 30 30 11 11 1e 9d 9d   cb
09a9 : 9d 9d 9d 9d 42 45 53 54   74
09b1 : 9f 1d 11 11 11 9d 9d 9d   f5
09b9 : 9d 9d 1e 4c 45 56 45 4c   eb
09c1 : 9d 9d 9d 9d 9d 11 11 11   11
09c9 : 1e 4c 49 56 45 53 9f 9d   d3
09d1 : 9d 9d 9d 11 34 1e 11 11   61
09d9 : 9d 9d 42 4c 4f 43 4b 53   42
09e1 : 11 9d 9d 9d 9d 9d 9f 31   83
09e9 : 33 11 11 9d 9d 9d 1e 42   61
09f1 : 4f 4e 55 53 13 00 00 00   58
09f9 : 00 00 00 00 00 00 00 00   fa
0a01 : 00 00 00 00 02 00 00 0a   36
0a09 : 01 00 08 02 00 08 02 80   96
0a11 : 00 00 60 00 00 90 00 00   ae
0a19 : 20 00 00 00 00 10 00 00   ba
0a21 : 68 00 00 24 00 00 08 00   2e
0a29 : 00 00 00 00 00 00 00 00   2a
0a31 : 00 00 00 00 a0 00 00 00   3c
0a39 : 00 00 00 00 00 00 00 00   3a
0a41 : 00 01 40 00 06 90 40 0a   cc
0a49 : 41 90 19 03 80 19 01 90   6f
0a51 : 04 00 60 00 00 64 00 00   91
0a59 : 24 00 10 10 00 64 00 00   a7
0a61 : e9 00 00 69 00 00 19 00   dc
0a69 : 00 04 00 00 00 00 00 00   6c
0a71 : 00 00 00 00 82 00 00 00   9a
0a79 : 00 00 00 00 00 00 05 00   8e
0a81 : 00 3a 40 40 19 93 90 ea   fd
0a89 : 93 a4 fa 4e a4 e9 0e 90   ea
0a91 : e9 03 a4 3b 00 e4 0c 14   0c
0a99 : e9 00 64 fb 00 69 0c 03   9c
0aa1 : aa 40 03 b6 40 00 6a 40   31
0aa9 : 00 3a c0 00 0d 00 00 00   c7
0ab1 : 00 00 00 00 90 00 00 00   bb
0ab9 : 00 00 00 00 40 00 00 15   e8
0ac1 : 40 36 5d 90 fa 99 a4 e9   69
0ac9 : a6 a4 fa aa a9 e6 be 99   d5
0ad1 : ea 43 a9 f9 90 e5 38 65   ea
0ad9 : a9 3e aa a9 03 ea 65 00   9e
0ae1 : ee a4 03 a9 91 03 aa 90   14
0ae9 : 00 eb b0 00 3c c0 00 4c   6d
0af1 : 00 00 00 00 b9 00 00 00   8d
0af9 : 00 85 00 00 19 40 05 5a   18
0b01 : 50 1a 9a 90 da a6 54 e9   1f
0b09 : aa a4 fa 6a 65 ea aa a5   b5
0b11 : e6 ab 99 fa ae a9 3a aa   09
0b19 : b9 3e 66 a9 0f aa a9 0e   c9
0b21 : aa e5 0e 59 a4 0f aa 94   03
0b29 : 03 fa a4 00 3e b0 00 0f   5a
0b31 : c8 00 80 00 10 00 00 00   1b
0b39 : 00 00 00 00 00 00 00 00   3a
0b41 : 00 00 00 00 00 01 e0 00   cd
0b49 : 03 f0 00 07 f8 00 07 f8   43
0b51 : 00 07 f8 00 07 f8 00 03   51
0b59 : f0 00 01 e0 00 00 00 00   a6
0b61 : 00 00 00 00 00 00 00 00   62
0b69 : 00 00 00 00 00 00 00 00   6a
0b71 : 00 00 00 00 20 00 00 00   74
0b79 : 00 00 00 00 00 00 00 00   7a
0b81 : 00 00 00 00 00 00 c0 00   85
0b89 : 03 f0 00 07 f8 00 0f fc   ab
0b91 : 00 0f fc 00 07 f8 00 03   96
0b99 : f0 00 00 c0 00 00 00 00   a2
0ba1 : 00 00 00 00 00 00 00 00   a2
0ba9 : 00 00 00 00 00 00 00 00   aa
0bb1 : 00 00 00 00 00 00 00 00   b2
0bb9 : 00 00 00 03 ff f8 03 aa   43
0bc1 : a8 03 aa ac 03 aa ac 03   69
0bc9 : aa a8 03 ab a8 03 af e8   31
0bd1 : 03 af e8 03 af e8 03 ab   ec
0bd9 : a8 03 aa a8 03 ab a8 03   f9
0be1 : ab a8 03 ab a8 03 ab a8   b9
0be9 : 02 aa a8 00 00 00 00 00   6b
0bf1 : 00 00 00 00 00 00 00 00   f2
0bf9 : 00 00 00 00 00 00 00 00   fa
0c01 : 00 00 00 00 00 01 e0 00   8d
0c09 : 03 00 00 06 00 00 06 00   e5
0c11 : 00 06 00 00 00 00 00 00   15
0c19 : 00 00 00 00 00 00 00 00   1a
0c21 : 00 00 00 00 00 00 00 00   22
0c29 : 00 00 00 00 00 00 00 00   2a
0c31 : 00 00 00 00 00 1e 31 5b   9e
0c39 : 4c 45 56 45 4c 5d 9d 9d   c7
0c41 : 9d 9d 9d 11 9f 30 31 11   99
0c49 : 11 0d 1d 1d 1d 1d 1d 08   0b
0c51 : 44 45 53 49 47 4e 5e 53   3d
0c59 : 4f 55 4e 44 5e 47 52 41   5b
0c61 : 50 48 49 43 53 20 41 4e   68
0c69 : 44 0d 1d 1d 1d 1d 1d 1d   88
0c71 : 1d 1d 1d 1d 1d 1d 50 52   a8
0c79 : 4f 47 52 41 4d 20 42 59   ba
0c81 : 20 4f 4c 49 56 45 52 20   9e
0c89 : 4b 49 52 57 41 0d 11 1d   f3
0c91 : 1d 1d 1d 1d 1d 1d 1d 1d   91
0c99 : 1d 1d 1d 1d 1d 1d 5c 20   9c
0ca1 : 31 39 38 37 5e 31 39 38   29
0ca9 : 38 11 11 97 0d c5 c3 c3   37
0cb1 : c3 c3 20 c5 c3 c3 c3 c4   09
0cb9 : 20 20 c2 20 c2 20 20 20   8c
0cc1 : 20 20 c2 20 20 20 20 20   6a
0cc9 : c2 20 c5 c3 c3 c3 c4 20   33
0cd1 : c5 c3 c3 c3 c4 c2 20 20   04
0cd9 : 20 20 20 c2 12 81 c8 ca   50
0ce1 : cb 97 92 c2 20 20 c2 20   c3
0ce9 : c2 20 20 20 20 20 c2 20   16
0cf1 : 20 20 20 20 c2 20 c2 12   8a
0cf9 : 81 cf d0 cc 97 92 c2 20   89
0d01 : c2 20 20 20 c2 c2 20 20   e2
0d09 : 20 20 20 c2 c3 c3 c3 c3   8b
0d11 : c4 20 c2 20 c2 20 20 20   88
```

```
0d19 : 20 20 c2 20 20 20 20 20   c2
0d21 : c2 20 c2 12 81 ce cd d4   56
0d29 : 97 92 c2 20 c2 20 20 20   ac
0d31 : c2 c2 20 20 20 20 20 c2   6a
0d39 : 20 20 20 20 c2 20 c2 20   ee
0d41 : c2 20 20 20 20 20 c2 20   6e
0d49 : 20 20 20 20 c2 20 c2 12   e2
0d51 : 81 d6 d5 d2 97 92 c2 20   66
0d59 : c2 20 20 20 c2 c7 c3 c3   38
0d61 : c3 c3 20 c2 20 20 20 20   2a
0d69 : c2 20 c2 20 c7 c3 c3 c3   21
0d71 : c3 20 c7 c3 c3 c3 c3 20   58
0d79 : c2 20 c7 c3 c3 c3 c6 20   6b
0d81 : c2 20 20 20 c2 aa bf bf   5f
0d89 : bf bf bf bf bf ab ff ff   69
0d91 : ff ff ff ff ff bf bf bf   0c
0d99 : bf bf bf bf ff ff ff ff   20
0da1 : ff ff ff ff ff aa bf bf   73
0da9 : bf bb bb be bf ab ff ff   66
0db1 : bf bb bb af bf ba bf be   81
0db9 : bb bb bf bf ff eb bf af   f7
0dc1 : bb bb bf ff ff aa bf bf   1d
0dc9 : be ba b8 b8 ba ab ff ff   32
0dd1 : af ab 8b 8b ab be be bf   d5
0dd9 : bb ba bb bf ff af af bf   94
0de1 : fb ab fb ff ff 2a 3a 3f   69
0de9 : 00 a2 a3 f3 00 aa ea ff   a3
0df1 : 00 a2 a3 f3 00 aa bf bf   7d
0df9 : bf bf bf be be ab ff fe   a7
0e01 : fe ff bf af af be bf bf   54
0e09 : bf bf bf bf ff af bf ff   0c
0e11 : bf bf bf bf ff 20 9a e5   cf
0e19 : 20 64 1c 20 7c 1a d8 a9   c6
0e21 : 3f 8d 02 dd a9 c4 8d 00   5a
0e29 : dd a9 cc 8d 88 02 a9 30   5f
0e31 : 8d 18 d0 a9 06 8d 22 d0   2b
0e39 : a9 01 8d 23 d0 a9 00 8d   a0
0e41 : 21 d0 8d 20 d0 8d 11 d0   91
0e49 : 20 64 1f a0 00 a9 30 99   b8
0e51 : 02 07 c8 c0 06 d0 f6 a2   29
0e59 : f0 bd 3f 08 9d 07 c0 ca   a3
0e61 : d0 f7 a2 51 bd 30 09 9d   bc
0e69 : 7f c1 ca d0 f7 a2 00 bd   a5
0e71 : f6 09 9d 00 c8 bd f5 0a   ba
0e79 : 9d ff c8 bd f6 0b 9d 00   3e
0e81 : ca e8 e0 ff d0 e9 a9 00   fa
0e89 : 8d fe c9 a2 41 bd bf c9   f1
0e91 : 85 02 4a 05 02 9d 3f ca   ea
0e99 : ca d0 f2 a9 00 8d 19 07   9c
0ea1 : a9 04 8d 79 14 a9 d8 8d   ec
0ea9 : 16 d0 a9 80 8d 92 d4 8d   7e
0eb1 : 8f d4 a9 00 8d 11 d0 20   fa
0eb9 : 10 1a 20 44 1d a9 17 8d   fe
0ec1 : 11 d0 a9 22 85 fe a9 71   c3
0ec9 : 85 fd 20 33 1d 20 0b 1f   f9
0ed1 : ad 01 dc 09 f0 c9 ff f0   96
0ed9 : 35 a0 04 a2 1c 20 83 1b   bb
0ee1 : 20 6e 1a 88 d0 f5 ad 53   ea
0ee9 : cd c9 32 90 19 ad 54 cd   25
0ef1 : c9 34 90 12 a9 30 8d 53   34
0ef9 : cd a9 31 8d 54 cd a9 22   37
0f01 : 85 fe a9 71 85 fd a2 ff   71
0f09 : 20 f9 1b 4c d1 0e a5 cb   32
0f11 : c9 40 f0 bc a9 00 8d 11   c1
0f19 : d0 ad 54 cd 8d 01 04 ad   db
0f21 : 53 cd 8d 02 04 a2 00 bd   cf
0f29 : 81 09 20 d2 ff e8 e0 75   47
0f31 : d0 f5 a9 30 8d 3f cd 8d   91
0f39 : 1d 07 ad 01 04 8d 2b ce   5c
0f41 : ad 02 04 8d 2a ce a9 80   63
0f49 : 8d 92 d4 8d 8f d4 a0 00   28
0f51 : b9 02 07 99 b2 cd c8 c0   3f
0f59 : 06 d0 f5 a9 39 8d 93 cf   68
0f61 : 8d 94 cf a0 a5 b1 fd 8d   3b
0f69 : 92 cf a9 13 20 d2 ff a9   9b
0f71 : 0b 8d 86 02 a2 17 a0 00   8a
0f79 : c0 00 d0 0d a9 0b 8d 86   45
0f81 : 02 a9 00 85 c7 a9 c2 d0   7f
0f89 : 23 c0 1f d0 0d a9 0b 8d   54
0f91 : 86 02 a9 00 85 c7 a9 c2   45
0f99 : d0 12 a9 08 8d 86 02 85   fe
0fa1 : c7 ad 9b d4 4d 04 dc 29   7b
0fa9 : 07 18 69 68 20 d2 ff 20   fd
0fb1 : a4 1a c8 c0 20 d0 c1 a9   8f
0fb9 : 0d 20 d2 ff ca d0 b7 a2   e2
0fc1 : 20 a9 43 9d ff cb 9d 97   3e
0fc9 : cf a9 0b 9d ff d7 9d 97   48
0fd1 : db ca d0 ed a2 44 8e 1f   c8
0fd9 : cc e8 8e 00 cc e8 8e b7   7b
0fe1 : cf e8 8e 98 cf a9 30 8d   01
0fe9 : 1a cf 8d 1b cf a9 00 8d   17
0ff1 : e8 06 a0 01 a2 00 20 6f   ae
0ff9 : 1c 98 48 ac e8 06 b1 fd   8a
1001 : d0 03 4c 24 11 85 02 29   82
1009 : 0f 09 08 8d ed 06 a5 02   fa
1011 : 29 f0 a0 00 c9 80 d0 20   ff
1019 : a9 64 8d e9 06 a9 65 8d   f3
1021 : ea 06 a9 66 8d ec 06 a9   f1
1029 : 67 8d eb 06 8a 48 a2 7a   7d
1031 : 20 97 1b 68 aa 4c c6 10   39
1039 : c9 40 d0 17 a9 68 8d e9   21
1041 : 06 a9 69 8d ea 06 a9 6a   82
1049 : 8d ec 06 a9 6b 8d eb 06   e2
1051 : 4c c6 10 c9 20 d0 17 a9   76
1059 : 6c 8d e9 06 a9 6d 8d ea   d9
1061 : 06 a9 6e 8d ec 06 a9 6f   0e
1069 : 8d eb 06 4c c6 10 c9 10   2b
1071 : d0 35 a9 70 8d eb 06 8d   c0
1079 : ea 06 a0 00 b1 fb c9 70   91
1081 : f0 0e c9 71 f0 0a [illegible] 71   02
1089 : 8d e9 06 8d ec 06 d0 14   a8
1091 : a9 70 8d e9 06 8d ec 06   9f
1099 : ad 9b d4 c9 d2 90 05 a9   9b
10a1 : 71 8d eb 06 4c c6 10 c9   63
10a9 : f0 d0 17 a9 72 8d e9 06   44
10b1 : a9 73 8d ea 06 a9 74 8d   6f
10b9 : ec 06 a9 75 8d eb 06 4c   aa
10c1 : c6 10 4c 24 11 a0 01 ad   9c
10c9 : e9 06 91 fb ad ed 06 20   3c
10d1 : 87 1a c8 ad ea 06 91 fb   6a
10d9 : ad ed 06 20 87 1a a0 29   21
10e1 : ad ec 06 91 fb ad ed 06   a9
10e9 : 20 87 1a a0 2a ad eb 06   33
10f1 : 91 fb ad ed 06 20 87 1a   5d
10f9 : e0 1c f0 1c a0 2b a9 ff   b1
1101 : 91 fb a9 08 20 87 1a 68   73
1109 : 48 c9 15 f0 16 a0 53 a9   a0
1111 : ff 91 fb a9 08 20 87 1a   e1
1119 : a0 52 a9 ff 91 fb a9 08   fc
1121 : 20 87 1a 68 a8 ee e8 06   4a
1129 : e8 e8 e0 1e f0 03 4c f7   c9
1131 : 0f c8 c8 c0 17 f0 03 4c   8c
1139 : f5 0f a9 43 8d b6 cf a9   aa
1141 : 0b 8d b6 db a0 00 b1 fd   09
1149 : 99 20 07 c8 c0 a5 d0 f6   38
1151 : a9 00 8d fd 06 8d fc 06   ea
1159 : a9 03 8d fe 06 20 a9 1a   03
1161 : a9 80 8d 1b d0 a0 a7 b1   25
1169 : fd 8d 29 d0 c8 b1 fd 09   b5
1171 : 03 29 fb 8d 04 d0 c8 b1   07
1179 : fd 09 01 29 f9 8d 05 d0   22
1181 : a9 00 8d 15 d0 8d 2e d0   04
1189 : a9 0f 8d 26 d0 a9 0b 8d   84
1191 : 25 d0 a9 3a 8d 1c d0 a9   20
1199 : 26 8d fa cf 8d ff cf a9   aa
11a1 : 24 8d f9 cf a9 17 8d 11   b0
11a9 : d0 a9 28 8d f8 cf a0 a6   e7
11b1 : b1 fd 8d 1e 07 a9 6f 8d   1f
11b9 : e5 06 8d e4 06 a9 00 8d   6a
11c1 : 13 07 8d 14 07 8d 15 07   7d
11c9 : 8d f6 06 a9 01 8d f3 06   e0
11d1 : 8d ff 06 8d 01 07 a9 14   a8
11d9 : 8d 09 07 8d 0b 07 8d 08   8e
11e1 : 07 a9 ff 8d 1c 07 20 10   09
11e9 : 1a a2 e1 20 f9 1b a9 ff   f0
11f1 : 8d 15 d0 20 64 1c a9 3a   83
11f9 : 8d 98 d4 a9 f4 8d 97 d4   00
1201 : a9 6b 8d 94 d4 a9 81 8d   11
1209 : 92 d4 a9 ff 8d 8f d4 a9   6c
1211 : 09 8d 81 d4 a9 1d 8d 19   c8
1219 : 03 a9 88 8d 18 03 ad e4   df
1221 : 06 f0 06 ee 05 d0 4c 2d   61
1229 : 12 ce 05 d0 ad e5 06 d0   c1
1231 : 06 ce 04 d0 4c 3f 12 c9   54
1239 : 6f f0 03 ee 04 d0 ad 05   46
1241 : d0 29 07 c9 01 f0 03 4c   dd
1249 : ce 13 ad 01 dc 29 04 d0   f5
1251 : 08 a9 00 8d e5 06 4c 6e   7c
1259 : 12 ad 01 dc 29 08 d0 08   44
1261 : a9 01 8d e5 06 4c 6e 12   4b
1269 : a9 6f 8d e5 06 ad 05 d0   6d
1271 : 4a 4a 4a 38 e9 06 8d e7   4f
1279 : 06 ad 04 d0 4a 4a 4a 38   02
1281 : e9 01 8d e6 06 ae e6 06   a8
1289 : 8e f0 06 ac e7 06 ad e4   d6
1291 : 06 f0 1a c8 8c f1 06 20   60
1299 : 6f 1c a0 00 b1 fb c9 c8   f2
12a1 : b0 36 a9 00 8d e4 06 20   2f
12a9 : 44 15 4c c5 12 88 8c f1   bf
12b1 : 06 20 6f 1c a0 00 b1 fb   ef
12b9 : c9 c8 b0 1c a9 01 8d e4   39
12c1 : 06 20 44 15 a9 26 8d fa   83
12c9 : cf a9 11 8d f3 06 a9 00   79
12d1 : 8d 10 ca a9 c0 8d 11 ca   a0
12d9 : ae e6 06 ac e7 06 8c f1   d6
12e1 : 06 ad e5 06 f0 21 e8 8e   d1
12e9 : f0 06 20 6f 1c ad e5 06   a5
12f1 : c9 6f f0 2d a0 00 b1 fb   1c
12f9 : c9 c8 b0 25 a9 00 8d e5   94
1301 : 06 20 44 15 4c ce 13 ca   e8
1309 : 8e f0 06 20 6f 1c a0 00   ef
1311 : b1 fb c9 c8 b0 0b a9 01   57
1319 : 8d e5 06 20 44 15 4c ce   da
1321 : 13 ad e5 06 c9 6f d0 03   a6
1329 : 4c ce 13 ae e6 06 ac e7   98
1331 : 06 ad e5 06 f0 07 ad e4   10
1339 : 06 f0 09 d0 2a ad e4 06   c3
1341 : f0 48 d0 69 e8 88 8e f0   a5
1349 : 06 8c f1 06 20 6f 1c a0   02
1351 : 00 b1 fb c9 c8 b0 76 a9   a1
1359 : 01 8d e4 06 a9 00 8d e5   b7
1361 : 06 20 44 15 4c ce 13 c8   44
1369 : e8 8e f0 06 8c f1 06 20   46
1371 : 6f 1c a0 00 b1 fb c9 c8   ca
1379 : b0 53 a9 00 8d e4 06 a9   a9
1381 : 00 8d e5 06 20 44 15 4c   93
1389 : ce 13 88 ca 8e f0 06 8c   fe
1391 : f1 06 20 6f 1c a0 00 b1   a6
1399 : fb c9 c8 b0 30 a9 01 8d   30
13a1 : e4 06 a9 01 8d e5 06 20   73
13a9 : 44 15 4c ce 13 ca c8 8e   2c
13b1 : f0 06 8c f1 06 20 6f 1c   5d
13b9 : a0 00 b1 fb c9 c8 b0 0d   05
13c1 : a9 00 8d e4 06 a9 01 8d   37
13c9 : e5 06 20 44 15 ad 04 d0   b2
13d1 : 8d 00 d0 18 69 08 8d 0e   bf
```

Listing 2. »Crillion+« ist eine ungepackte Version, die sich problemlos editieren läßt. Bitte mit dem MSE (Seite 159) eingeben.

```
13d9 : d0 ad 05 d0 8d 01 d0 c9  93
13e1 : e0 90 02 a9 00 18 69 08  36
13e9 : 8d 0f d0 ad fa cf 8d ff  4c
13f1 : cf ad 29 d0 09 08 ae 29  d9
13f9 : d0 e0 f7 d0 02 a9 01 e0  84
1401 : f3 d0 02 a9 0d e0 f4 d0  5f
1409 : 02 a9 0a 8d 27 d0 20 ed  69
1411 : f6 d0 19 a9 80 8d 92 d4  53
1419 : 20 ed f6 f0 fb 20 ed f6  72
1421 : d0 fb 20 ed f6 f0 fb a9  ef
1429 : 81 8d 92 d4 ad 0a 07 f0  d9
1431 : 10 ee 0b 07 ad 0b 07 8d  c6
1439 : 96 d4 cd 09 07 90 38 b0  05
1441 : 10 ce 0b 07 ad 0b 07 8d  c6
1449 : 96 d4 cd 09 07 b0 28 90  96
1451 : 00 ad 09 07 8d 08 07 ad  dc
1459 : 9b d4 29 7f c9 6f b0 f7  63
1461 : 8d 09 07 ee 09 07 ad 08  a2
1469 : 07 cd 09 07 90 04 a9 00  4a
1471 : f0 02 a9 01 8d 0a 07 a2  77
1479 : c8 20 f9 1b ce ff 06 d0  da
1481 : 15 ad 01 07 f0 10 a2 f3  90
1489 : 20 ab 1b a2 0c 8e ff 06  db
1491 : ad 00 07 8d 01 07 ce f3  1d
1499 : 06 d0 0f a9 25 8d fa cf  4b
14a1 : a9 01 8d 10 ca a9 e0 8d  c9
14a9 : 11 ca a2 00 20 1a 1b a2  4d
14b1 : 01 20 1a 1b a2 02 20 1a  9b
14b9 : 1b ad 1c 07 c9 ff f0 08  03
14c1 : ce 1c 07 d0 03 4c 5f 18  b9
14c9 : ac f6 06 f0 4b ce f8 06  ab
14d1 : d0 22 ac f9 06 ae fa 06  ea
14d9 : 84 fb 86 fc 20 00 1c a9  62
14e1 : 00 8d f6 06 8d 03 d0 8d  76
14e9 : 0d d0 8d 10 d0 a9 0f 8d  75
14f1 : 26 d0 d0 24 c0 01 f0 1a  44
14f9 : c0 02 f0 10 c0 03 f0 06  ec
1501 : ce 02 d0 4c 16 15 ee 02  58
1509 : d0 4c 16 15 ee 03 d0 4c  0a
1511 : 16 15 ce 03 d0 20 de 1a  84
1519 : ad 10 07 c9 01 d0 03 20  ac
1521 : ec 19 c9 02 d0 0c ad 0f  8f
1529 : 07 18 69 56 8d 0f 07 8d  ea
1531 : 88 d4 c9 03 d0 09 ce 11  a9
1539 : 07 ad 11 07 8d 88 d4 ea  82
1541 : 4c 15 12 ae f0 06 f0 0f  93
1549 : e0 1f f0 0b ac f1 06 f0  ab
1551 : 06 c0 17 f0 02 d0 03 4c  e7
1559 : 15 18 ce f0 06 ad f0 06  ea
1561 : 4a 8d ee 06 ce f1 06 ad  de
1569 : f1 06 4a 8d ef 06 0a 0a  0d
1571 : 0a 0a 38 ed ef 06 18 6d  b6
1579 : ee 06 8d f2 06 aa ee f0  7f
1581 : 06 ee f1 06 bd 20 07 29  87
1589 : f0 c9 40 d0 2e bd 20 07  e7
1591 : 29 07 8d 29 d0 0a 0a 18  7c
1599 : 69 14 8d 88 d4 8d 0c 07  79
15a1 : 8d 0d 07 8d 0e 07 a9 00  e8
15a9 : 8d 8b d4 a9 09 8d 8c d4  3f
15b1 : a9 11 8d 8b d4 a9 01 8d  71
15b9 : 10 07 60 c9 10 d0 03 4c  ca
15c1 : fd 17 c9 80 f0 03 4c 5c  dd
15c9 : 16 bd 20 07 29 07 09 f0  78
15d1 : cd 29 d0 f0 03 4c 15 18  9c
15d9 : a9 00 9d 20 07 20 57 1a  f1
15e1 : 20 a9 1c a0 04 ad 13 07  f9
15e9 : f0 09 a0 06 ad 14 07 f0  c0
15f1 : 02 a0 08 20 c3 1a a0 00  d9
15f9 : ad 13 07 f0 09 a0 01 ad  05
1601 : 14 07 f0 02 a0 02 a9 01  d8
1609 : 99 13 07 99 16 07 a9 20  a1
1611 : 99 fb cf a9 0c 99 2a d0  a9
1619 : a2 7a 20 ab 1b ad 1b cf  a1
1621 : c9 30 d0 0c ad 1a cf c9  36
1629 : 30 d0 05 a9 32 8d 1c 07  46
1631 : a9 00 8d 8b d4 a9 09 8d  89
1639 : 8c d4 a9 81 8d 8b d4 a9  a6
1641 : 00 8d 88 d4 8d 0f 07 a9  85
1649 : 02 8d 10 07 ac 1e 07 a2  14
1651 : 07 20 e0 1b 88 d0 f8 20  37
1659 : 59 1b 60 c9 f0 f0 03 4c  cc
1661 : f6 17 ad f6 06 f0 03 4c  ba
1669 : 15 18 bd 20 07 29 07 09  e6
1671 : f0 cd 29 d0 f0 03 4c 15  2f
1679 : 18 09 08 8d ed 06 29 07  8b
1681 : 0a 0a 0a 0a 18 69 0a 8d  64
1689 : 12 07 ad f0 06 cd e6 06  1f
1691 : f0 09 ad f1 06 cd e7 06  2a
1699 : f0 01 60 ae f2 06 ad e6  dc
16a1 : 06 cd f0 06 90 43 d0 03  f7
16a9 : 4c 28 17 b0 00 ad ee 06  1a
16b1 : d0 03 4c 15 18 ca 86 02  af
16b9 : bd 20 07 f0 03 4c 15 18  7d
16c1 : ae f2 06 bd 20 07 a6 02  fa
16c9 : 9d 20 07 ae f2 06 a9 00  14
16d1 : 9d 20 07 20 57 1a 20 a9  5e
16d9 : 1c a0 00 20 c3 1a a9 04  05
16e1 : 8d f6 06 ce ee 06 4c b1  f8
16e9 : 17 ad ee 06 c9 0e d0 03  aa
16f1 : 4c 15 18 e8 86 02 bd 20  9b
16f9 : 07 f0 03 4c 15 18 ae f2  75
1701 : 06 bd 20 07 a6 02 9d 20  00
1709 : 07 ae f2 06 a9 00 9d 20  36
1711 : 07 20 57 1a 20 a9 1c a0  42
1719 : 00 20 c3 1a a9 03 8d f6  34
1721 : 06 ee ee 06 4c b1 17 ad  25
1729 : e7 06 cd f1 06 90 42 b0  14
1731 : 00 ad ef 06 d0 03 4c 15  45
1739 : 18 8a 38 e9 0f aa 86 02  46
1741 : bd 20 07 f0 03 4c 15 18  05
1749 : ae f2 06 bd 20 07 a6 02  82
1751 : 9d 20 07 ae f2 06 a9 00  9c
1759 : 9d 20 07 20 57 1a 20 a9  e6
1761 : 1c a0 00 20 c3 1a a9 01  87
1769 : 8d f6 06 ce ef 06 4c b1  90
1771 : 17 ad ef 06 c9 0a d0 03  52
1779 : 4c 15 18 8a 18 69 0f aa  06
1781 : bd 20 07 f0 03 4c 15 18  45
1789 : 86 02 ae f2 06 bd 20 07  f7
1791 : a6 02 9d 20 07 ae f2 06  61
1799 : a9 00 9d 20 07 20 57 1a  b1
17a1 : 20 a9 1c a0 00 20 c3 1a  f5
17a9 : a9 02 8d f6 06 ee ef 06  39
17b1 : 20 57 1a a4 fb 8c f9 06  b0
17b9 : a6 fc 8e fa 06 a9 10 8d  e9
17c1 : f8 06 ad 12 07 8d 11 07  99
17c9 : 8d 88 d4 a9 01 8d 26 d0  bb
17d1 : a9 27 8d f9 cf ad ed 06  de
17d9 : 29 f7 8d 28 d0 20 de 1a  24
17e1 : a9 00 8d 8b d4 a9 90 8d  57
17e9 : 8c d4 a9 15 8d 8b d4 a9  c8
17f1 : 03 8d 10 07 60 c9 20 d0  16
17f9 : 1b 4c 88 1d a9 00 8d 84  da
1801 : d4 8d 86 d4 a9 02 8d 85  c4
1809 : d4 a9 09 8d 81 d4 a9 81  0e
1811 : 8d 84 d4 60 a9 00 8d 84  fb
1819 : d4 8d 86 d4 a9 02 8d 85  dc
1821 : d4 a9 08 8d 81 d4 a9 11  05
1829 : 8d 84 d4 60 a0 00 a9 08  fa
1831 : 85 fb a9 c2 85 fc a9 00  5d
1839 : 91 fb a5 fb 18 69 08 85  a9
1841 : fb a5 fc 69 00 85 fc a5  e7
1849 : fb c9 00 90 e9 a5 fc c9  8e
1851 : c8 90 e3 a2 0a 20 f9 1b  6e
1859 : c8 c0 08 d0 d1 60 20 7c  37
1861 : 1a a9 80 8d 92 d4 a9 00  98
1869 : 8d 15 d0 8d 05 d0 8d 07  82
1871 : d0 8d 0b d0 8d 09 d0 8d  64
1879 : 03 d0 8d 0d d0 a9 21 8d  e3
1881 : ed 1e a9 f1 8d ec 1e ce  7c
1889 : ef 1e ee f2 1e a9 01 8d  f0
1891 : fd 06 a9 f7 8d 97 d4 a2  29
1899 : ff 20 bc 1f a9 00 8d 8b  a3
18a1 : d4 8d 8c d4 a9 b8 8d 8d  ab
18a9 : d4 a9 03 8d 12 07 a9 ff  c4
18b1 : 8d 0f 07 8d 81 d4 a9 15  c9
18b9 : 8d 8b d4 a9 2a 8d 98 d4  91
18c1 : ad 19 07 d0 15 ad 2a ce  dc
18c9 : c9 32 d0 4a ad 2b ce c9  2c
18d1 : 35 d0 43 a9 58 8d 19 07  d9
18d9 : d0 05 a9 42 8d 19 07 a9  f0
18e1 : 04 8d 79 14 a9 22 85 fe  4c
18e9 : a9 71 85 fd ad 04 dc 29  2d
18f1 : 3f f0 f9 cd 1a 07 f0 f4  68
18f9 : cd 1b 07 f0 ef c9 1a b0  4b
1901 : eb aa ad 1a 07 8d 1b 07  47
1909 : 8e 1a 07 f0 0c ca f0 09  71
1911 : 20 6e 1a 4c 0e 19 20 6e  80
1919 : 1a a0 0a 8c 00 04 a2 19  74
1921 : ca d0 fd a2 07 20 e0 1b  52
1929 : 20 59 1b ce 00 04 d0 ee  d8
1931 : a2 f3 20 ab 1b ce 12 07  c9
1939 : d0 19 a9 02 8d 12 07 ad  21
1941 : 0f 07 f0 09 ce 0f 07 8d  ce
1949 : 88 d4 8d 96 d4 ad 04 dc  f6
1951 : 8d 88 d4 ad 00 07 d0 c1  0c
1959 : a9 14 8d 8b d4 20 2d 18  14
1961 : ad 19 07 d0 08 a2 f3 20  1c
1969 : 83 1b 4c 7d 19 a9 12 8d  7f
1971 : 2a ce a9 0e 8d 2b ce a9  ef
1979 : 04 8d 2c ce ee ef 1e ce  ad
1981 : f2 1e ad 19 07 c9 58 d0  d3
1989 : 55 a9 01 8d 88 d4 a9 36  e7
1991 : 8d 0c 07 8d 0d 07 8d 0e  f3
1999 : 07 a2 ff 20 f9 1b a9 00  14
19a1 : 8d 8b d4 8d 97 d4 a9 0d  bb
19a9 : 8d 8c d4 a9 09 8d 8d d4  c3
19b1 : a9 63 8d 81 d4 a9 15 8d  a9
19b9 : 8b d4 a2 03 86 02 a0 b7  22
19c1 : a2 04 20 f9 1b 20 ec 19  45
19c9 : a9 01 8d fd 06 88 d0 f0  e0
19d1 : c6 02 d0 ea a9 80 8d 8b  16
19d9 : d4 a2 ff 20 f9 1b a9 21  64
19e1 : 8d ed 1e a9 e9 8d ec 1e  1d
19e9 : 4c 47 0f ad 0d 07 18 69  8f
19f1 : 06 8d 0d 07 8d 88 d4 cd  ee
19f9 : 0e 07 90 12 ad 0e 07 18  89
1a01 : 69 06 8d 0e 07 ad 0c 07  af
1a09 : 8d 0d 07 8d 88 d4 60 a2  86
1a11 : 80 ad 9b d4 45 a2 2d 04  0f
1a19 : dc 29 aa c9 aa f0 f2 49  fe
1a21 : ff 85 02 a9 ff 25 02 9d  05
1a29 : 3f c6 ca d0 e4 a2 00 bd  77
1a31 : 10 08 9d 10 c2 e8 e0 30  06
1a39 : d0 f5 a2 00 bd 86 0d 9d  2c
1a41 : 20 c3 e8 e0 90 d0 f5 a2  46
1a49 : 08 a9 00 9d ff c0 a9 ff  86
1a51 : 9d f7 c7 ca d0 f3 ad ee  76
1a59 : 06 0a aa ad ef 06 0a a8  6d
1a61 : 20 6f 1c 60 ad 9b d4 29  a9
1a69 : 07 18 69 c8 60 a5 fd 18  4b
1a71 : 69 aa 85 fd a5 fe 69 00  48
1a79 : 85 fe 60 a9 fe 8d 19 03  91
1a81 : a9 c1 8d 18 03 60 48 a5  11
1a89 : fc 18 69 0c 85 fc 68 91  72
1a91 : fb a5 fc 38 e9 0c 85 fc  b4
1a99 : 60 48 a9 ff 38 e9 01 d0  00
1aa1 : fb 68 60 48 a9 23 d0 f4  d2
1aa9 : 78 a9 1e 8d 15 03 a9 c7  cf
1ab1 : 8d 14 03 a9 81 8d 1a d0  cd
1ab9 : ad 11 d0 29 7f 8d 11 d0  92
1ac1 : 58 60 ad ef 06 0a 0a 0a  a0
```

```
1ac9 : 0a 18 69 3c 99 03 d0 ad  12
1ad1 : ee 06 0a 0a 0a 0a 18 69  aa
1ad9 : 1a 99 02 d0 60 a9 29 8d  6e
1ae1 : fe cf a9 00 8d 2d d0 ad  12
1ae9 : 1b d0 09 40 8d 1b d0 ad  07
1af1 : 02 d0 18 69 08 8d 0c d0  4d
1af9 : 90 0b a9 40 8d 10 d0 ce  bb
1b01 : 0c d0 ce 0c d0 ad 03 d0  d3
1b09 : 18 69 08 8d 0d d0 c9 e3  d0
1b11 : 90 06 ce 0d d0 ce 0d d0  53
1b19 : 60 de 16 07 d0 39 a9 04  d4
1b21 : 9d 16 07 bd 13 07 f0 2f  ce
1b29 : c9 02 f0 0b bd fb cf c9  1f
1b31 : 24 f0 04 fe fb cf 60 a9  c1
1b39 : 02 9d 13 07 a9 09 9d 16  35
1b41 : 07 bd fb cf c9 20 f0 04  89
1b49 : de fb cf 60 a9 00 9d 13  5c
1b51 : 07 8a 0a aa 9d 07 d0 60  8b
1b59 : a0 00 b9 b2 cd d9 3a cd  ee
1b61 : 90 0f f0 02 b0 05 c8 c0  cd
1b69 : 06 d0 ef a0 28 88 d0 fd  ed
1b71 : 60 a0 00 b9 3a cd 99 b2  36
1b79 : cd 99 02 07 c8 c0 06 d0  c1
1b81 : f2 60 fe 38 cd bd 38 cd  b1
1b89 : c9 3a d0 09 a9 30 9d 38  c8
1b91 : cd ca 20 83 1b 60 fe a1  30
1b99 : ce bd a1 ce c9 3a d0 09  4c
1ba1 : a9 30 9d a1 ce ca 20 97  f1
1ba9 : 1b 60 86 02 bd a1 ce c9  8e
1bb1 : 30 90 1c c9 3a b0 18 a9  46
1bb9 : 01 8d 00 07 de a1 ce bd  13
1bc1 : a1 ce c9 2f d0 18 a9 39  08
1bc9 : 9d a1 ce ca 4c ad 1b a9  36
1bd1 : 00 8d 00 07 a6 02 20 97  a3
1bd9 : 1b a9 20 8d 91 cf 60 20  dc
1be1 : 83 1b ad 3b cd cd 1d 07  92
1be9 : f0 0d 8d 1d 07 ad a2 ce  6d
1bf1 : c9 37 b0 03 ee a2 ce 60  e2
1bf9 : 20 9a 1a ca d0 fa 60 a9  00
1c01 : 72 a0 29 91 fb ad ed 06  31
1c09 : 20 87 1a a0 2a a9 73 91  68
1c11 : fb ad ed 06 20 87 1a a0  07
1c19 : 51 a9 74 91 fb ad ed 06  7f
1c21 : 20 87 1a a0 52 a9 75 91  0b
1c29 : fb ad ed 06 20 87 1a a0  1f
1c31 : 53 b1 fb c9 c8 90 09 a9  1d
1c39 : ff 91 fb a9 08 20 87 1a  09
1c41 : a0 7b b1 fb c9 c8 90 09  c2
1c49 : a9 ff 91 fb a9 08 20 87  40
1c51 : 1a a0 7a b1 fb c9 c8 90  e2
1c59 : 09 a9 ff 91 fb a9 08 20  d6
1c61 : 87 1a 60 a2 16 a9 00 9d  4c
1c69 : 80 d4 ca d0 f8 60 98 85  20
1c71 : fb a9 00 85 fc 06 fb 26  2e
1c79 : fc 06 fb 26 fc 98 18 65  fc
1c81 : fb 85 fb a5 fc 69 00 85  18
1c89 : fc 06 fb 26 fc 06 fb 26  88
1c91 : fc 06 fb 26 fc a5 fc 18  75
1c99 : 69 cc 85 fc 8a 65 fb 85  38
1ca1 : fb a5 fc 69 00 85 fc 60  bc
1ca9 : a0 29 20 65 1a 91 fb a9  04
1cb1 : 08 20 87 1a 20 65 1a a0  c5
1cb9 : 2a 91 fb a9 08 20 87 1a  b4
1cc1 : a0 51 20 65 1a 91 fb a9  30
1cc9 : 08 20 87 1a a0 52 20 65  ee
1cd1 : 1a 91 fb a9 08 20 87 1a  bc
1cd9 : a0 53 b1 fb c9 ff d0 0a  02
1ce1 : 20 65 1a 91 fb a9 08 20  da
1ce9 : 87 1a a0 7a b1 fb c9 ff  17
1cf1 : d0 0a 20 65 1a 91 fb a9  ed
1cf9 : 08 20 87 1a a0 7b b1 fb  db
1d01 : c9 ff d0 0a 20 65 1a 91  f8
1d09 : fb a9 08 20 87 1a a0 00  ab
1d11 : 20 1f 1d a0 01 20 1f 1d  e4
1d19 : a0 28 20 1f 1d 60 b1 fb  4d
1d21 : c9 c8 b0 0d c9 47 90 09  47
1d29 : 98 18 69 29 a8 a9 ff 91  48
1d31 : fb 60 a5 c5 c9 40 d0 fa  56
1d39 : 60 20 33 1d a5 cb c9 40  7a
1d41 : f0 fa 60 20 44 e5 a0 10  e1
1d49 : a2 07 18 20 f0 ff a2 00  13
1d51 : bd 36 0c 20 d2 ff e8 e0  c3
1d59 : fc d0 f5 a0 03 84 02 a2  f0
1d61 : 00 bd 32 0d 20 d2 ff e8  d8
1d69 : e0 2c d0 f5 c6 02 d0 ef  f2
1d71 : a2 00 bd 5e 0d 20 d2 ff  6c
1d79 : e8 e0 28 d0 f5 a2 cc 8e  ba
1d81 : 67 cf e8 8e 8e cf 60 20  05
1d89 : 7c 1a a2 18 bd 08 22 9d  9e
1d91 : 80 d4 ca d0 f7 20 a9 1a  a3
1d99 : a9 81 8d 84 d4 8d 92 d4  a4
1da1 : 8d 8b d4 a9 04 8d 15 d0  01
1da9 : 8d 0d d0 8d 07 d0 8d 09  e2
1db1 : d0 8d 0b d0 8d 03 d0 8d  74
1db9 : 1c d0 8d fd 06 a9 02 8d  31
1dc1 : fe 06 a9 07 8d 29 d0 8d  8e
1dc9 : 2e d0 a9 02 8d 25 d0 a9  a3
1dd1 : 01 8d 26 d0 a9 00 8d 1b  43
1dd9 : d0 8d fb 06 a9 24 8d fa  17
1de1 : cf 8d fb 06 ce fb 06 ad  77
1de9 : fa cf c9 20 f0 25 ce fa  ab
1df1 : cf a2 22 20 f9 1b ad fa  c3
1df9 : cf cd fb 06 d0 e9 ee fa  7c
1e01 : cf a2 22 20 f9 1b ad fa  d3
1e09 : cf c9 24 d0 f1 ce fb 06  71
1e11 : 4c e5 1d a9 00 8d 04 d0  ea
1e19 : ce a2 ce a2 ff 20 f9 1b  5f
1e21 : a9 ab 8d 6d c3 a2 23 20  cf
1e29 : f9 1b a9 ab 8d 6e c3 a2  30
1e31 : 3c 20 f9 1b a9 8b [illegible] 6e  69
1e39 : c3 a2 23 20 f9 1b a9 8b  50
1e41 : 8d 6d c3 a2 e1 20 f9 1b  07
1e49 : a2 ff 20 f9 1b 20 2d 18  ca
1e51 : ad a2 ce c9 30 f0 03 4c  6b
1e59 : 47 0f a9 c8 8d 16 d0 a2  bd
1e61 : ff 20 f9 1b a9 01 8d fd  27
1e69 : 06 a9 2a 8d 98 d4 a9 00  57
1e71 : 8d 92 d4 a9 f7 8d 97 d4  a5
1e79 : a9 69 8d 94 d4 a9 81 8d  88
1e81 : 92 d4 a9 ff 8d 0f 07 8d  70
1e89 : 8f d4 a0 00 a9 0c 85 d6  69
1e91 : 84 d3 20 10 e5 a2 00 bd  f8
1e99 : fd 21 20 d2 ff e8 e0 0b  6a
1ea1 : d0 f5 a2 09 20 9a 1a ce  12
1ea9 : 0f 07 ce 0f 07 ad 0f 07  fa
1eb1 : 8d 96 d4 ca d0 ee c8 c0  41
1eb9 : 0d d0 d1 a9 80 8d 92 d4  40
1ec1 : 20 3a 1d 4c fa 1f ae 19  65
1ec9 : d0 8e 19 d0 30 07 ae 0d  51
1ed1 : dc 58 4c 31 ea ad fd 06  33
1ed9 : f0 28 ce fe 06 d0 23 a9  37
1ee1 : 02 8d fe 06 ae fc 06 ee  f3
1ee9 : fc 06 bd e9 21 8d 20 d0  35
1ef1 : 8d 21 d0 d0 0d a9 00 8d  96
1ef9 : fd 06 8d fc 06 a9 06 8d  dd
1f01 : 22 d0 a9 01 8d 12 d0 4c  5b
1f09 : 7e ea a9 2f 8d 98 d4 a9  91
1f11 : ea 8d ce 0e 8d cf 0e 8d  e2
1f19 : d0 0e a9 f7 8d 97 d4 a9  96
1f21 : 00 8d 81 d4 8d 84 d4 8d  4e
1f29 : 92 d4 a9 20 8d 88 d4 a9  57
1f31 : f0 8d 93 d4 a9 fa 8d 94  39
1f39 : d4 a9 15 8d 92 d4 a0 ff  2b
1f41 : 20 a2 1f d0 18 8c 96 d4  7e
1f49 : 88 d0 f5 20 a2 1f d0 0d  3b
1f51 : 8c 96 d4 c8 c0 70 d0 f3  31
1f59 : a9 1e 8d 96 d4 a9 80 8d  ff
1f61 : 92 d4 60 a2 08 a0 01 20  94
1f69 : ba ff a9 05 a2 6c a0 22  82
1f71 : 20 bd ff a9 00 20 d5 ff  fd
1f79 : 90 08 c9 05 f0 0e c9 00  c7
1f81 : f0 0a 20 ca 21 ad 82 05  6b
1f89 : c9 30 f0 14 a2 00 a0 00  d6
1f91 : b9 47 22 9d 05 04 e8 c8  d0
1f99 : c0 13 d0 f4 e0 a0 90 ee  e9
1fa1 : 60 a9 ff 38 e9 1d 8d 8f  ba
1fa9 : d4 a2 04 20 f9 1b c9 3c  ec
1fb1 : b0 f1 a5 cb c9 3c f0 02  83
1fb9 : c9 40 60 20 64 1c a9 00  8c
1fc1 : 8d 97 d4 a9 0b 8d 93 d4  99
1fc9 : a9 15 8d 92 d4 a0 18 a9  b8
1fd1 : ff a2 02 8e 88 d4 20 9a  59
1fd9 : 1a 38 e9 0f 8d 8f d4 c9  a8
1fe1 : 14 b0 f3 a9 00 8d 8f d4  d4
1fe9 : 8d 88 d4 a2 22 20 f9 1b  85
1ff1 : 88 d0 dc a9 f7 8d 97 d4  41
1ff9 : 60 a2 00 8e 37 03 bd 3a  73
2001 : cd 9d 78 05 e8 e0 06 d0  ab
2009 : f5 a2 00 a0 00 ee 37 03  be
2011 : 8e 36 03 b9 78 05 dd 05  e4
2019 : 04 90 46 f0 02 b0 09 e8  b1
2021 : c8 c0 06 d0 ee 4c 62 20  00
2029 : ae 36 03 a0 85 b9 05 04  09
2031 : 99 18 04 88 c0 ff f0 08  c8
2039 : cc 36 03 90 03 4c 2e 20  7f
2041 : a0 00 b9 78 05 9d 05 04  b8
2049 : e8 c8 c0 06 d0 f4 8e 38  e6
2051 : 03 a0 00 a9 20 9d 05 04  e5
2059 : e8 c8 c0 0d d0 f5 4c 71  48
2061 : 20 e8 c8 c0 13 d0 fa e0  a5
2069 : 96 90 a0 a9 63 8d 37 03  2a
2071 : a2 00 bd 21 22 20 d2 ff  15
2079 : e8 e0 1e d0 f5 a9 00 8d  3b
2081 : 34 03 8d 35 03 a0 31 84  44
2089 : 02 a5 02 38 e9 2e 0a 38  8e
2091 : e9 01 aa a0 05 18 20 f0  2d
2099 : ff ae 34 03 bd 3f 22 8d  d6
20a1 : 86 02 a5 02 20 d2 ff a9  be
20a9 : 5b 20 d2 ff a2 04 a9 1d  f4
20b1 : 20 d2 ff ca d0 f8 ae 35  8d
20b9 : 03 a0 00 bd 05 04 20 d2  5b
20c1 : ff ee 35 03 e8 c8 c0 06  c9
20c9 : d0 f1 a2 06 a9 1d 20 d2  a5
20d1 : ff ca d0 f8 ae 35 03 a0  6a
20d9 : 00 bd 05 04 20 d2 ff ee  f0
20e1 : 35 03 e8 c8 c0 0d d0 f1  87
20e9 : ee 34 03 e6 02 a5 02 c9  78
20f1 : 39 d0 96 ad 37 03 c9 63  67
20f9 : d0 03 4c a1 21 a2 17 a0  57
2101 : 0c 18 20 f0 ff a2 00 bd  d0
2109 : 5a 22 20 d2 ff e8 e0 10  c2
2111 : d0 f5 ae 37 03 bd 3e 22  ca
2119 : 8d 86 02 a0 17 ad 37 03  40
2121 : 0a 18 69 03 aa 18 20 f0  c0
2129 : ff a2 00 86 c6 8e 80 05  37
2131 : 20 e4 ff c9 41 90 07 c9  45
2139 : 5c b0 03 4c aa 21 c9 14  3b
2141 : d0 0b ae 80 05 f0 e9 ce  70
2149 : 80 05 20 d2 ff c9 2e d0  57
2151 : 05 a9 5b 4c aa 21 c9 20  a6
2159 : d0 03 4c aa 21 c9 0d d0  49
2161 : cf a0 0b a2 17 18 20 f0  2c
2169 : ff a2 16 a9 20 20 d2 ff  c2
2171 : ca d0 f8 a2 08 a0 01 20  00
2179 : ba ff a9 07 a2 6a a0 22  c2
2181 : 20 bd ff a2 04 86 fc a2  82
2189 : 05 86 fb a9 fb a2 af a0  da
2191 : 04 20 d8 ff 90 04 c9 05  36
2199 : f0 09 20 ca 21 4c a4 21  b9
```

Listing 2. (Fortsetzung)

```
21a1 : 20 3a 1d 20 44 e5 4c 9c   08
21a9 : 0e 48 ae 80 05 e0 0d d0   c4
21b1 : 03 4c 31 21 20 d2 ff ad   3f
21b9 : 38 03 18 6d 80 05 aa 68   d2
21c1 : 9d 05 04 ee 80 05 4c 31   83
21c9 : 21 a9 08 85 ba 20 b4 ff   f1
21d1 : a9 6f 85 b9 20 96 ff a2   c6
21d9 : 00 20 a5 ff 9d 82 05 e8   26
21e1 : c9 0d d0 f5 20 ab ff 60   44
21e9 : 0b 0c 0f 01 0f 0c 0b 00   5c
21f1 : 0b 0c 0f 03 0d 07 0d 03   6a
21f9 : 0f 0c 0b 00 99 20 47 41   0b
2201 : 4d 45 20 4f 56 45 52 00   bc
2209 : 02 00 00 00 0c 00 00 03   d2
2211 : 00 00 00 0c 00 00 02 00   9b
2219 : 00 00 0c 00 00 32 f7 1f   cc
2221 : 93 9f 11 1d 1d 1d 1d 1d   d5
2229 : 1d 1d 1d 1d 1d 1d 1d 1d   29
2231 : 54 48 45 20 54 4f 50 20   40
2239 : 45 49 47 48 54 5d 07 0d   64
2241 : 03 0f 0c 0c 0b 0b 30 30   7b
2249 : 35 30 30 30 20 44 52 5b   cd
2251 : 4b 4e 4f 58 20 20 20 20   66
2259 : 20 9f 45 4e 54 45 52 20   5d
2261 : 59 4f 55 52 20 4e 41 4d   16
2269 : 45 40 3a 54 4f 50 20 38   50
2271 : 00 00 00 00 00 00 00 00   72
2279 : 00 00 00 00 00 00 00 00   7a
2281 : 00 00 00 00 00 00 86 00   9c
2289 : 00 00 00 00 00 00 00 00   8a
2291 : 00 00 00 00 86 86 86 00   48
2299 : 00 00 00 00 00 00 00 00   9a
22a1 : 00 00 84 84 84 84 84 00   d1
22a9 : 00 00 00 00 00 00 00 00   aa
22b1 : 84 84 84 84 84 84 44 00   a6
22b9 : 00 00 00 00 00 00 22 22   86
22c1 : 22 22 22 22 22 22 22 00   7d
22c9 : 00 00 00 00 00 00 84 84   e5
22d1 : 84 84 84 84 84 00 00 00   91
22d9 : 00 00 00 00 00 00 84 84   f5
22e1 : 84 84 84 00 00 00 00 00   c9
22e9 : 00 00 00 00 00 00 86 86   11
22f1 : 86 00 00 00 00 00 00 00   78
22f9 : 00 00 00 00 00 00 86 00   14
2301 : 00 00 00 00 00 00 00 00   02
2309 : 00 00 00 00 00 00 00 00   0a
2311 : 00 00 00 00 00 35 51 06   0c
2319 : de de 00 00 00 00 00 00   67
2321 : 00 00 00 00 00 00 00 f2   07
2329 : 00 00 44 82 82 82 82 82   d6
2331 : 82 82 82 82 82 82 82 00   2b
2339 : 00 15 15 15 15 15 15 15   24
2341 : 15 15 15 15 15 15 00 00   c3
2349 : 00 00 00 00 00 00 00 00   4a
2351 : 00 00 00 00 f4 00 00 46   2d
2359 : 84 84 84 84 84 84 84 84   58
2361 : 84 84 84 84 00 00 15 15   57
2369 : 15 15 15 15 15 15 15 15   69
2371 : 15 15 15 00 00 00 00 00   56
2379 : 00 00 00 00 00 00 00 00   7a
2381 : 00 00 00 00 42 86 86 86   01
2389 : 86 86 86 86 86 86 86 86   88
2391 : 86 00 00 15 15 22 15 15   9b
2399 : 22 15 15 22 22 22 15 15   81
23a1 : 00 00 00 00 00 00 00 00   a2
23a9 : 00 00 00 00 00 00 00 00   aa
23b1 : 00 00 00 00 00 00 00 00   b2
23b9 : 00 00 00 00 00 00 00 35   24
23c1 : 46 06 1c 50 22 86 22 86   07
23c9 : 22 86 22 86 22 86 22 86   74
23d1 : 22 86 22 00 86 00 86 00   41
23d9 : 86 00 86 00 86 00 86 00   83
23e1 : 86 00 00 00 00 86 00 86   a9
23e9 : 00 86 00 86 00 86 00 00   31
```

```
23f1 : 00 00 86 00 00 00 86 00   ad
23f9 : 86 00 86 00 00 00 86 00   3b
2401 : 00 86 00 86 00 00 00 86   22
2409 : 00 00 00 86 00 86 00 00   0e
2411 : 86 00 86 00 86 00 00 00   a1
2419 : 86 00 86 00 86 00 00 86   b6
2421 : 00 86 00 00 00 86 00 00   99
2429 : 00 86 00 86 00 00 86 00   57
2431 : 00 00 86 00 86 00 86 00   55
2439 : 00 00 86 00 00 00 00 86   e8
2441 : 00 86 00 86 00 86 00 86   96
2449 : 00 00 00 00 86 00 86 00   cc
2451 : 86 00 86 00 86 00 86 00   fb
2459 : 86 00 00 86 00 86 00 86   f1
2461 : 00 86 00 86 00 86 00 86   b6
2469 : 00 37 27 06 26 58 86 f5   bb
2471 : 86 86 86 86 86 f5 22 86   5a
2479 : 86 86 86 86 86 86 f5 f5   15
2481 : f5 86 f5 f5 f5 f5 f5 f5   c8
2489 : f5 86 f5 22 86 86 86 86   47
2491 : 86 86 86 86 86 f5 86 86   0c
2499 : 86 f5 f5 86 f5 f5 f5 f5   3b
24a1 : f5 86 f5 86 f5 86 f5 86   a0
24a9 : 86 86 86 86 86 86 86 f5   87
24b1 : 86 f5 86 86 86 f5 86 f5   c2
24b9 : 86 f5 f5 f5 f5 86 f5 86   ee
24c1 : f5 86 f5 f5 f5 86 f5 86   ae
24c9 : 86 86 86 f5 86 86 22 f5   04
24d1 : 86 f5 22 86 86 f5 86 86   ea
24d9 : f5 86 f5 f5 86 f5 f5 86   4a
24e1 : f5 f5 f5 f5 f5 86 86 f5   a6
24e9 : 86 86 f5 86 86 f5 86 86   40
24f1 : 86 86 86 f5 22 86 f5 f5   35
24f9 : 86 f5 86 f5 f5 f5 f5 f5   ad
2501 : f5 86 f5 f5 22 f5 86 86   6e
2509 : 86 86 86 f5 22 86 86 86   b0
2511 : 86 86 86 38 1c 06 ca d5   4c
2519 : 00 00 00 00 00 f2 00 00   b1
2521 : 22 00 00 00 82 82 82 00   8a
2529 : 00 00 00 00 12 00 00 00   4b
2531 : 00 00 00 82 82 44 00 00   cc
2539 : 00 f2 00 12 84 00 00 00   3d
2541 : 00 00 82 82 82 12 12 12   57
2549 : 00 12 12 12 12 12 12 12   37
2551 : 12 12 12 12 00 00 00 00   33
2559 : 00 00 00 00 00 00 00 00   5a
2561 : 00 00 00 00 84 00 84 84   c5
2569 : 84 00 84 84 84 00 84 84   02
2571 : 84 00 00 84 00 84 00 84   b3
2579 : 00 84 00 84 00 82 00 84   69
2581 : 00 00 84 00 84 84 84 00   21
2589 : 84 84 84 00 82 82 84 00   bf
2591 : 00 84 00 00 00 84 00 84   01
2599 : 00 84 00 82 00 84 00 00   50
25a1 : 84 00 00 00 84 00 84 84   89
25a9 : 84 00 82 82 84 00 00 00   67
25b1 : 00 00 00 00 00 00 00 00   b2
25b9 : 00 00 00 00 00 37 32 02   40
25c1 : 26 50 87 00 87 00 87 00   88
25c9 : f7 87 f7 00 87 00 87 00   18
25d1 : 87 00 87 00 87 00 f7 00   93
25d9 : f7 00 f7 00 87 00 87 00   65
25e1 : 87 00 87 00 87 00 f7 87   b2
25e9 : f7 00 87 00 87 00 87 00   59
25f1 : 87 00 87 00 f7 00 f7 00   ba
25f9 : f7 00 87 00 87 00 87 00   69
2601 : 87 00 87 00 f7 87 f7 00   06
2609 : 87 00 87 00 87 00 87 00   09
2611 : 87 00 f7 00 f7 00 f7 00   f6
2619 : 87 00 87 00 87 00 87 00   19
2621 : 87 00 f7 87 f7 00 87 00   35
2629 : 87 00 87 00 87 00 87 00   29
2631 : f7 00 f7 00 f7 00 87 00   c4
2639 : 87 00 87 00 87 00 87 00   39
```

```
2641 : f7 87 f7 00 87 00 87 00   90
2649 : 87 00 87 00 87 00 f7 00   0b
2651 : f7 00 f7 00 87 00 87 00   dd
2659 : 87 00 87 00 87 00 f7 87   2a
2661 : f7 00 87 00 87 00 87 37   3f
2669 : 2b 07 30 cb 22 46 00 00   f2
2671 : 00 00 00 00 00 00 00 00   72
2679 : 86 00 86 22 22 00 00 00   07
2681 : 00 00 00 00 00 00 86 00   9c
2689 : 46 00 22 22 22 00 00 00   be
2691 : 00 00 00 00 00 00 86 00   ac
2699 : 86 22 22 22 22 00 00 00   1f
26a1 : 00 00 00 00 86 00 86 00   24
26a9 : 22 22 00 22 22 00 00 00   43
26b1 : 00 00 00 00 86 00 86 00   34
26b9 : 22 22 22 22 22 00 00 00   db
26c1 : 00 00 86 00 86 00 22 22   98
26c9 : 22 00 22 22 22 00 00 00   da
26d1 : 00 00 86 00 86 22 22 22   b9
26d9 : 22 22 00 22 22 00 00 00   73
26e1 : 86 00 86 00 22 00 22 22   f8
26e9 : 22 22 22 22 22 00 00 00   0b
26f1 : 86 00 86 22 22 22 00 22   d5
26f9 : 22 00 22 22 22 00 00 00   0a
2701 : 00 00 22 22 22 22 22 22   ce
2709 : 22 22 22 22 22 00 00 00   2b
2711 : 00 34 93 02 ca d1 00 00   8c
2719 : 00 00 44 00 00 00 00 00   2b
2721 : 00 00 85 85 83 00 00 00   6c
2729 : 00 44 00 43 00 00 00 00   b4
2731 : 00 85 85 83 00 00 00 00   c6
2739 : 13 13 13 13 13 00 00 00   2e
2741 : 85 85 85 00 00 00 00 13   10
2749 : 00 13 00 13 00 00 00 85   40
2751 : 83 85 00 00 00 00 13 00   e3
2759 : 13 00 13 00 00 00 83 83   46
2761 : 83 00 00 00 00 13 00 13   a3
2769 : 00 13 00 00 00 83 85 85   30
2771 : 13 22 22 f4 00 00 13 00   09
2779 : 13 00 00 00 83 85 83 00   ff
2781 : 00 00 00 00 00 00 00 13   a8
2789 : 00 00 00 83 83 83 87 87   7b
2791 : 47 87 00 00 00 00 00 00   9c
2799 : 00 00 85 85 85 00 00 84   0d
27a1 : 00 00 00 00 00 00 00 00   a2
27a9 : 00 85 83 83 45 84 84 84   51
27b1 : 00 00 00 00 00 00 00 00   b2
27b9 : 85 83 84 39 3f 04 83 89   7d
27c1 : 00 00 00 00 00 00 00 00   c2
27c9 : 00 00 00 00 00 00 00 00   ca
27d1 : 00 00 00 00 00 00 00 85   dd
27d9 : 00 00 00 00 00 85 00 00   06
27e1 : 00 85 00 00 00 00 00 00   a4
27e9 : 00 85 00 85 85 00 85 85   d6
27f1 : 85 00 85 00 00 00 00 00   d8
27f9 : 85 00 13 13 00 00 00 85   b1
2801 : 00 00 00 00 00 00 00 85   0d
2809 : 00 00 00 00 f5 f5 f5 f5   dc
2811 : f5 f5 f5 f5 f5 f5 f5 f5   10
2819 : f5 f5 00 00 00 00 00 00   09
2821 : 00 00 00 00 00 00 00 00   22
2829 : 00 00 13 00 13 00 13 00   6c
2831 : 13 00 13 00 13 00 13 00   87
2839 : 00 13 00 13 00 13 00 13   e4
2841 : 00 13 00 13 00 13 00 00   c6
2849 : 47 00 44 00 42 00 44 00   d7
2851 : 46 00 43 00 45 00 00 13   e3
2859 : 87 13 84 13 82 13 84 13   e6
2861 : 86 13 83 13 85 35 7d 05   b6
2869 : e3 46 82 f6 00 00 00 f6   dd
2871 : 00 00 00 f6 00 00 00 f6   3e
2879 : 42 00 f6 00 f6 00 f6 00   c4
2881 : f6 00 f6 00 f6 00 f6 00   80
2889 : 00 f6 00 f6 00 f6 00 f6   89
```

```
2891 : 00 f6 00 f6 00 f6 00 00  a3
2899 : f6 00 f6 00 f6 00 f6 00  98
28a1 : f6 00 f6 00 f6 00 00 f6  b2
28a9 : 00 f6 00 f6 00 f6 00 f6  a9
28b1 : 00 f6 00 f6 00 00 f6 00  e7
28b9 : f6 00 f6 00 f6 00 f6 00  b8
28c1 : f6 00 f6 00 00 f6 00 f6  1b
28c9 : 00 f6 00 f6 00 f6 00 f6  c9
28d1 : 00 f6 00 00 f6 00 f6 00  98
28d9 : f6 00 f6 00 f6 00 f6 00  d8
28e1 : f6 00 00 f6 00 f6 00 f6  5c
28e9 : 00 f6 00 f6 00 f6 00 f6  e9
28f1 : 00 00 f6 00 f6 00 f6 00  fa
28f9 : f6 00 f6 00 f6 00 f6 00  f8
2901 : 00 00 00 f6 00 00 00 f6  ce
2909 : 00 00 00 f6 00 00 00 36  55
2911 : ff 06 1a 7b 84 84 84 84  91
2919 : 84 15 84 00 84 15 84 84  55
2921 : 84 84 84 87 87 87 87 87  db
2929 : 15 15 00 15 15 87 87 87  26
2931 : 87 87 82 82 82 82 82 82  b8
2939 : 82 00 86 86 86 86 86 86  f1
2941 : 86 85 85 85 85 85 85 15  61
2949 : 00 15 85 85 85 85 85 85  8b
2951 : 83 83 83 83 83 83 86 00  56
2959 : 82 83 83 83 83 83 83 86  5e
2961 : 86 86 86 86 86 86 00 82  3e
2969 : 82 82 82 82 82 82 15 15  d8
2971 : 15 15 15 15 15 f2 15 15  60
2979 : 15 15 15 15 15 15 22 46  10
2981 : 22 47 22 45 00 43 22 42  9f
2989 : 22 44 22 15 15 00 00 00  4a
2991 : 00 00 00 00 00 00 00 00  92
2999 : 00 00 15 15 00 00 00 00  81
29a1 : 00 00 00 00 00 00 00 00  a2
29a9 : 00 15 15 00 00 00 00 00  79
29b1 : 00 00 00 00 00 00 00 00  b2
29b9 : 15 39 2b 02 83 d3 83 00  5b
29c1 : 00 00 00 00 83 00 00 83  01
29c9 : 00 00 00 00 00 00 00 00  ca
29d1 : 00 83 00 83 83 00 00 00  3c
29d9 : 00 00 00 00 00 00 83 00  e8
29e1 : 83 00 83 00 00 83 00 00  61
29e9 : 00 83 00 00 00 83 00 00  c7
29f1 : 00 00 00 00 83 00 83 83  3f
29f9 : 43 00 83 00 83 00 83 00  63
2a01 : 83 83 00 83 00 00 00 83  bd
2a09 : 00 00 00 83 00 83 00 83  9d
2a11 : 00 00 83 00 00 00 83 00  00
2a19 : 00 f2 83 00 83 00 83 00  b9
2a21 : 00 83 00 00 00 83 00 00  ff
2a29 : 00 83 00 83 00 83 00 00  77
2a31 : 83 83 00 00 83 00 00 83  b5
2a39 : 83 00 83 00 83 00 00 00  d5
2a41 : 00 00 00 83 00 00 83 00  c0
2a49 : 00 00 00 83 00 00 83 00  c8
2a51 : 00 83 83 00 00 00 00 00  f4
2a59 : 00 00 83 00 00 83 83 00  64
2a61 : 00 83 22 39 4f 02 1c 5a  fd
2a69 : 00 83 83 83 83 83 83 83  e5
2a71 : 00 83 83 83 83 83 83 00  e6
2a79 : 83 83 83 83 83 83 83 83  78
2a81 : 83 83 83 00 83 83 00 83  02
2a89 : 83 83 83 00 83 83 83 83  18
2a91 : 00 83 83 83 83 00 83 83  f1
2a99 : 83 83 83 83 83 83 83 83  98
2aa1 : 83 83 83 83 00 45 45 45  01
2aa9 : 45 45 45 45 45 45 45 45  a9
2ab1 : 45 45 45 00 11 11 11 11  ea
2ab9 : 11 11 11 11 11 11 11 11  b9
2ac1 : 11 11 00 f4 00 00 00 f4  e3
2ac9 : 00 00 00 f4 00 00 00 f4  52
2ad1 : 11 00 00 f4 00 f4 00 f4  13
2ad9 : 00 f4 00 f4 00 f4 00 11  bc
2ae1 : 43 00 00 f4 00 00 00 f4  ad
2ae9 : 00 00 00 f4 00 00 11 00  cc
2af1 : 00 f4 00 f4 00 f4 00 f4  9c
2af9 : 00 f4 00 f4 00 11 00 f4  84
2b01 : 00 00 00 f4 00 00 00 f4  8a
2b09 : 00 00 00 f4 11 36 30 04  34
2b11 : e1 ba 14 00 14 00 14 00  e6
2b19 : 14 00 14 00 14 00 14 00  c4
2b21 : 14 00 f5 85 f5 85 f5 85  d2
2b29 : f5 85 f5 85 f5 85 f5 00  72
2b31 : 14 00 14 00 14 00 14 00  dc
2b39 : 14 00 14 00 14 00 14 00  e4
2b41 : f5 00 f5 00 f5 00 f5 00  eb
2b49 : f5 00 f5 00 f5 00 14 00  6b
2b51 : 14 00 14 00 14 00 14 00  fc
2b59 : 14 00 14 00 14 00 f5 00  8c
2b61 : f5 00 f5 00 f5 00 f5 00  0b
2b69 : f5 00 f5 00 14 00 14 00  6d
2b71 : 14 00 14 00 14 00 14 00  1c
2b79 : 14 00 14 00 f5 00 f5 00  ca
2b81 : f5 00 f5 00 f5 00 f5 00  2b
2b89 : f5 00 14 00 14 00 14 00  15
2b91 : 14 00 14 00 14 00 14 00  3c
2b99 : 14 00 f5 85 f5 85 f5 85  4a
2ba1 : f5 85 f5 85 f5 85 f5 00  ea
2ba9 : 14 00 14 00 14 00 14 00  54
2bb1 : 14 00 14 00 14 00 14 36  c8
2bb9 : d0 05 dc a8 22 22 22 22  58
2bc1 : 22 22 22 22 22 22 22 22  c1
2bc9 : 22 22 22 00 00 00 00 85  90
2bd1 : 00 85 00 85 00 85 00 00  71
2bd9 : 00 00 00 00 00 00 00 00  da
2be1 : 00 00 00 00 00 00 00 00  e2
2be9 : 00 00 00 00 00 00 00 00  ea
2bf1 : 00 00 00 00 00 00 00 00  f2
2bf9 : 00 00 00 00 00 00 00 00  fa
2c01 : 00 00 00 00 00 00 00 00  02
2c09 : 00 00 00 00 00 00 00 00  0a
2c11 : 00 00 00 00 00 00 85 85  33
2c19 : 85 85 85 85 85 85 85 85  18
2c21 : 85 85 85 85 85 85 00 85  0a
2c29 : 00 85 85 85 85 85 85 85  a3
2c31 : 85 00 85 85 85 85 00 85  58
2c39 : 85 85 85 85 85 85 85 00  2d
2c41 : 85 00 85 85 85 85 85 85  7e
2c49 : 85 85 85 85 85 85 85 85  48
2c51 : 85 85 00 00 00 00 00 00  99
2c59 : 00 00 00 00 00 00 00 00  5a
2c61 : 00 36 2b 05 98 8c 00 13  fc
2c69 : 13 13 13 13 13 13 13 13  69
2c71 : 13 13 13 13 13 00 86 22  c5
2c79 : 86 86 86 86 86 86 86 86  78
2c81 : 86 86 86 86 00 86 86 86  18
2c89 : 86 86 86 86 86 86 22 86  f7
2c91 : 86 86 86 00 86 86 86 86  c0
2c99 : 86 86 86 86 86 86 86 22  d0
2ca1 : 86 86 00 86 86 86 86 86  ff
2ca9 : 22 86 86 86 86 86 86 86  44
2cb1 : 86 00 86 86 86 86 86 86  6d
2cb9 : 86 86 86 86 86 86 86 22  f0
2cc1 : 00 86 86 86 86 86 86 86  3a
2cc9 : 22 86 86 86 86 86 86 00  57
2cd1 : 86 86 86 22 86 86 86 86  44
2cd9 : 86 86 86 86 86 86 00 13  d7
2ce1 : 13 13 13 13 13 13 13 13  e1
2ce9 : 13 13 13 13 13 00 46 86  04
2cf1 : 46 86 86 86 86 86 46 86  af
2cf9 : 46 46 86 46 00 00 00 00  cd
2d01 : 00 f6 00 00 00 00 00 00  7d
2d09 : 00 00 00 38 19 05 ed dc  3c
2d11 : f2 82 f2 f2 f2 f2 f2 f2  d8
2d19 : f2 f2 f2 00 00 f2 f2 00  a4
2d21 : f2 82 82 82 82 82 82 f2  71
2d29 : 82 82 f2 00 00 f2 f2 f2  f2
2d31 : 82 f2 f2 f2 f2 00 f2 f2  28
2d39 : 82 00 82 f2 f2 f2 f2 82  52
2d41 : f2 f2 f2 82 f2 00 f2 f2  9a
2d49 : f2 f2 82 f2 00 f2 82 f2  3b
2d51 : 22 f2 00 82 f2 f2 f2 f2  b5
2d59 : f2 82 f2 f2 82 f2 f2 f2  19
2d61 : f2 f2 f2 f2 f2 00 f2 f2  c8
2d69 : 82 f2 00 82 f2 f2 f2 f2  2d
2d71 : f2 f2 f2 f2 f2 f2 f2 82  8f
2d79 : f2 f2 82 82 82 82 00 82  16
2d81 : 82 f2 f2 f2 00 f2 00 f2  15
2d89 : f2 82 82 82 42 f2 f2 f2  1b
2d91 : f2 f2 82 f2 00 f2 00 f2  79
2d99 : 82 f2 82 82 00 f2 f2 82  ee
2da1 : 82 82 82 f2 82 f2 f2 f2  d5
2da9 : f2 82 82 00 82 82 82 f2  a9
2db1 : f2 82 82 82 f2 38 34 02  9b
2db9 : 7b 79 00 00 22 00 00 00  13
2dc1 : 46 00 00 22 00 00 00 22  90
2dc9 : 00 00 00 42 00 00 00 22  56
2dd1 : 00 00 43 00 00 00 47 00  bf
2dd9 : 00 00 00 00 44 00 00 00  1e
2de1 : 00 00 00 45 00 00 00 00  8a
2de9 : 00 00 00 22 00 00 00 00  2e
2df1 : 00 00 22 00 00 00 13 13  ec
2df9 : 13 13 13 13 13 13 13 13  f9
2e01 : 13 13 13 00 00 00 00 f6  51
2e09 : 00 00 00 00 f4 00 00 00  59
2e11 : 00 f2 00 00 13 00 00 00  bc
2e19 : 13 13 00 00 00 13 13 00  9b
2e21 : 00 00 13 13 13 00 13 13  ec
2e29 : 13 13 00 13 13 13 13 00  3e
2e31 : 13 13 87 13 87 87 13 83  1a
2e39 : 13 83 83 13 85 13 85 85  63
2e41 : 13 87 13 f7 87 13 83 13  21
2e49 : f3 83 13 85 13 f5 85 13  90
2e51 : 87 87 87 87 13 83 83 83  d1
2e59 : 83 13 85 85 85 85 13 39  bb
2e61 : 5d 05 f3 78 00 00 00 00  4d
2e69 : 00 00 00 00 00 00 00 00  6a
2e71 : 00 00 00 00 f3 f3 f3 00  20
2e79 : 00 f3 f3 f3 00 f3 00 f3  76
2e81 : f3 00 00 f3 f3 f3 f3 f3  89
2e89 : 00 00 f3 f3 00 f3 f3 f3  5c
2e91 : 00 00 f3 f3 f3 f3 00 f3  d3
2e99 : f3 f3 f3 00 f3 f3 00 00  62
2ea1 : 00 00 f3 f3 00 00 00 00  1d
2ea9 : f3 f3 00 f3 f3 00 f3 f3  0b
2eb1 : 00 f3 00 f3 00 f3 00 f3  b1
2eb9 : 00 00 00 f3 f3 00 f3 f3  2f
2ec1 : f3 f3 00 f3 f3 f3 f3 00  db
2ec9 : f3 00 f3 f3 00 f3 00 00  d7
2ed1 : f3 00 f3 f3 f3 f3 00 00  1e
2ed9 : f3 00 f3 f3 f3 00 f3 f3  3f
2ee1 : f3 f3 f3 f3 f3 f3 f3 00  f8
2ee9 : f3 f3 f3 00 00 00 00 00  d3
2ef1 : 00 00 00 00 00 00 00 00  f2
2ef9 : 00 00 86 46 86 86 86 86  27
2f01 : 86 86 46 46 86 86 86 86  e8
2f09 : 86 35 d0 03 2a 3e 83 22  a5
2f11 : 83 83 15 83 83 22 83 83  6a
2f19 : 83 22 83 83 83 83 83 83  68
2f21 : 22 15 83 83 83 83 22 83  03
2f29 : 83 83 83 22 00 00 00 00  93
2f31 : 15 00 00 00 00 00 00 00  47
2f39 : 00 00 00 00 00 00 00 00  3a
2f41 : 00 00 00 00 00 00 00 00  42
2f49 : 00 00 00 00 00 00 15 15  c8
2f51 : 15 15 15 15 15 00 00 00  2a
2f59 : 00 00 00 00 00 15 84 84  1d
2f61 : 00 00 f3 15 00 00 00 00  01
```

Listing 2. (Fortsetzung)

```
2f69 : 00 00 00 00 15 84 f3 00  af
2f71 : 00 00 15 00 00 00 00 15  e1
2f79 : 15 00 15 15 15 15 f3 f3  28
2f81 : 00 15 15 15 00 15 00 00  9d
2f89 : 00 00 00 00 00 00 00 00  8a
2f91 : 15 00 00 00 00 83 15 83  1e
2f99 : 83 15 15 15 15 15 15 15  07
2fa1 : 83 83 83 22 83 15 83 83  01
2fa9 : 15 44 83 83 83 83 83 83  9b
2fb1 : 22 83 83 38 3c 03 a0 a4  24
2fb9 : 44 00 00 00 00 00 00 00  fe
2fc1 : 00 00 00 00 00 00 00 47  50
2fc9 : 46 13 13 13 13 13 13 13  fc
2fd1 : 13 13 13 13 13 00 00 00  c6
2fd9 : 00 00 00 00 00 00 00 00  da
2fe1 : 00 00 00 00 00 00 00 00  e2
2fe9 : 87 87 87 87 84 87 87 87  b8
2ff1 : 87 87 87 87 00 00 00 87  1e
2ff9 : f2 f2 22 f2 f2 22 22 22  58
3001 : 22 22 22 00 00 00 87 87  ea
3009 : 87 87 87 87 87 87 87 87  08
3011 : 87 87 00 00 00 00 00 00  5c
3019 : 00 00 00 00 00 00 00 00  1a
3021 : 00 87 87 87 87 87 87 87  99
3029 : 87 87 87 87 87 87 87 00  19
3031 : 22 22 22 22 22 22 f2 22  74
3039 : 22 22 22 22 22 87 00 47  26
3041 : 87 87 87 87 87 87 87 87  40
3049 : 87 87 87 87 87 00 00 00  df
3051 : 00 00 00 00 00 00 00 00  52
3059 : 00 00 00 00 00 36 2f 07  d6
3061 : 1a df 22 22 22 22 22 22  38
3069 : 22 22 22 22 22 22 22 22  69
3071 : 22 00 00 00 00 00 00 00  94
3079 : 00 00 00 00 00 00 00 00  7a
3081 : 00 00 00 00 00 00 00 00  82
3089 : 00 00 00 00 00 00 00 00  8a
3091 : 16 16 16 16 00 00 00 16  27
3099 : 00 00 16 00 00 00 00 16  4b
30a1 : 00 00 16 00 00 00 16 00  7f
30a9 : 00 16 00 00 00 00 84 00  c7
30b1 : 00 84 00 00 00 84 00 84  21
30b9 : 00 00 00 00 00 84 00 00  de
30c1 : 84 00 00 00 84 84 00 00  b2
30c9 : 00 00 00 00 84 00 00 84  1b
30d1 : 00 00 00 84 00 84 00 00  86
30d9 : 00 00 00 46 00 00 86 00  bc
30e1 : 00 00 86 00 00 86 00 00  b7
30e9 : 00 00 86 86 86 86 00 86  05
30f1 : 00 86 00 00 86 00 86 00  b7
30f9 : 00 00 00 00 00 00 00 00  fa
3101 : 00 00 00 00 00 00 00 35  6c
3109 : 6c 04 83 89 15 15 15 15  02
3111 : 15 15 15 15 15 15 15 15  11
3119 : 15 15 15 46 46 46 46 46  03
3121 : 46 87 87 87 46 87 87 87  cb
3129 : 87 87 87 87 87 87 87 87  28
3131 : 87 87 87 46 87 87 87 87  08
3139 : 87 87 87 87 87 87 87 87  38
3141 : 46 87 46 87 15 15 15 15  46
3149 : 00 00 00 00 00 00 00 00  4a
3151 : 00 00 87 00 00 00 00 87  42
3159 : 87 87 87 87 87 87 46 87  53
3161 : 46 87 15 f7 f7 00 87 87  5c
3169 : 87 87 87 87 87 46 87 46  dc
3171 : 87 15 00 00 00 46 46 46  5b
3179 : 46 46 46 87 46 87 46 87  2e
3181 : 15 00 00 15 00 00 46 87  61
3189 : 87 87 87 46 87 46 87 15  71
3191 : 00 15 47 00 00 46 87 87  4d
3199 : 87 87 46 87 87 87 15 00  6f
31a1 : 15 47 00 00 46 87 87 87  28
31a9 : 87 46 87 87 87 15 00 00  47
31b1 : 00 39 2c 06 ed dc 00 00  df
31b9 : 87 15 43 00 00 15 82 00  4e
31c1 : 00 15 00 00 00 00 00 00  4c
31c9 : 15 84 00 00 15 00 00 00  72
31d1 : f6 00 00 00 f6 15 15 15  5e
31d9 : 15 f5 15 15 15 15 15 15  49
31e1 : 15 f6 15 00 46 00 85 00  31
31e9 : 00 00 15 00 00 00 15 00  83
31f1 : 00 00 00 00 00 15 00 00  9a
31f9 : 00 f5 00 00 00 f5 00 00  a4
3201 : 85 15 f6 15 15 15 f5 15  6d
3209 : 15 15 f5 15 15 15 f5 15  c5
3211 : 00 00 00 15 00 00 00 15  de
3219 : 00 00 00 15 00 00 00 00  bc
3221 : 00 00 15 86 00 00 15 00  8c
3229 : 00 00 15 00 00 00 15 00  c3
3231 : 15 15 15 15 f5 15 87 15  09
3239 : 15 15 15 42 15 82 00 82  d1
3241 : 15 00 00 00 f3 00 00 00  96
3249 : f4 00 00 00 44 82 00 15  c0
3251 : 00 00 00 47 00 44 84 15  99
3259 : 86 84 46 39 d4 05 5c 74  aa
3261 : 42 87 87 87 87 00 87 87  df
3269 : 87 87 87 14 00 00 00 46  a5
3271 : 00 00 00 00 00 00 00 00  72
3279 : 00 00 82 00 00 00 45 87  3e
3281 : 87 87 87 00 87 87 87 87  90
3289 : 87 14 00 00 00 14 14 14  34
3291 : 14 14 00 14 14 14 14 14  8c
3299 : 14 00 00 00 86 86 86 86  71
32a1 : 86 00 86 86 86 86 86 14  79
32a9 : 00 00 00 00 00 00 00 00  aa
32b1 : 00 00 00 00 00 00 14 00  02
32b9 : 00 00 00 00 00 00 00 00  ba
32c1 : 00 00 00 00 00 14 00 00  62
32c9 : 00 00 00 00 00 00 00 00  ca
32d1 : 00 00 00 00 14 14 86 14  f6
32d9 : 00 00 00 00 00 00 00 00  da
32e1 : 00 00 00 14 82 82 82 00  aa
32e9 : 00 00 00 00 00 00 00 00  ea
32f1 : 00 00 14 82 f2 82 22 22  57
32f9 : 22 22 22 22 22 22 22 22  f9
3301 : 22 14 82 82 82 39 4a 07  47
3309 : a7 4e ea 41 4e 00 ff 00  9f
```

Listing 2. (Schluß)

```
10 N1$="CRILL-EDI"                                  <085>
15 U=8:D=0                                          <093>
60 PRINT"{CLR}BITTE CRILL-EDI /{2SPACE}DIS
   K IN LAUFWERK 8{3SPACE,DOWN}EINLEGEN UN
   D TASTE DRUECKEN"                                <154>
65 POKE 198,0:WAIT 198,1                            <037>
80 N$=N1$:P=676                                     <138>
110 V=16: Q=32: F=256: N$=LEFT$(N$,V)               <147>
120 H$="0123456789ABCDEF": Z$=CHR$(0)               <002>
130 T$=RIGHT$(STR$(D),1): OPEN 1,U,15,"I"+
    T$: GOSUB 620                                   <148>
140 T$=T$+":"+N$: OPEN 2,U,2,T$+",P,R"              <012>
150 INPUT#1,E,M$,J,K: IF E=0 THEN 240               <091>
160 CLOSE 1: CLOSE 2: PRINT "{DOWN,SPACE}P
    ROGRAMM:{2SPACE}";N$;                           <000>
170 IF E=62 THEN PRINT " NICHT GEFUNDEN":
    GOTO 200                                        <104>
180 IF E=64 THEN PRINT " FALSCHER FILETYP"
    : GOTO 200                                      <117>
190 PRINT "{DOWN,SPACE}DISK READ-ERROR";E:
     END                                            <173>
200 FOR J=1 TO 2000: NEXT: GOTO 70                  <112>
210 J=INT(C/V): K=C-J*V                             <122>
220 M$=M$+MID$(H$,J+1,1)+MID$(H$,K+1,1)             <144>
230 RETURN                                          <032>
240 GET#2,A$,B$: IF ST THEN 190                     <023>
250 CLOSE 2: CLOSE 1: M$=""                         <191>
260 A=ASC(A$+Z$): B=ASC(B$+Z$)                      <168>
270 C=B: GOSUB 210: C=A: GOSUB 210                  <039>
280 PRINT "{DOWN}DIE STARTADRESSE VON ";N$
    ;" IST: "                                       <178>
290 PRINT "{DOWN,SPACE}HEXADEZIMAL: ";M$;"
     DEZIMAL: ";B*F+A                               <245>
300 PRINT "UND WIRD GEAENDERT AUF DEZIMAL:
    ";P                                             <116>
340 B=INT(P/F): A=P-B*F: P$=CHR$(A)+CHR$(B
    ): M$=""                                        <133>
350 C=B: GOSUB 210: C=A: GOSUB 210                  <119>
360 PRINT "{DOWN,SPACE}HEX:{2SPACE}";M$;"
    DEZIMAL: ";B*F+A                                <216>
390 J=LEN(N$): IF J=V THEN 410                      <210>
400 FOR X=J+1 TO V:N$=N$+CHR$(160):NEXT             <171>
410 OPEN 1,U,15: OPEN 2,U,2,"#"                     <007>
420 GOSUB 620: T=18: S=1                            <010>
430 PRINT#1,"U1:";2;D;T;S: GOSUB 620                <166>
440 PRINT#1,"B-P:";2;0: GET#2,A$,B$                 <161>
450 T=ASC(A$+Z$): S=ASC(B$+Z$): H=2                 <102>
460 PRINT#1,"B-P:";2;H: GET#2,T$                    <156>
470 C=ASC(T$+Z$): IF C<>130 THEN 510                <234>
480 GET#2,A$,B$: F$="": FOR X=1 TO V                <068>
490 GET#2,T$: F$=F$+T$: NEXT                        <163>
500 IF F$=N$ THEN 530                               <241>
510 H=H+Q: IF H<F THEN 460                          <192>
520 GOTO 430                                        <012>
530 A=ASC(A$+Z$): B=ASC(B$+Z$)                      <184>
540 PRINT#1,"U1:";2;D;A;B: GOSUB 620                <106>
550 PRINT#1,"B-P:";2;2: PRINT#2,P$;                 <112>
560 PRINT#1,"U2:";2;D;A;B: GOSUB 620                <255>
570 CLOSE 2: GOSUB 620: CLOSE 1                     <250>
610 PRINT "{CLR}FERTIG! CRILL-EDI{4SPACE}K
    ANN NUN GESTARTET{SPACE,DOWN}WERDEN":E
    ND                                              <158>
620 INPUT#1,E,M$,J,K:IF E=0 THEN RETURN             <207>
630 PRINT "{DOWN,SPACE}ERROR: ";E;M$;J;K            <114>
640 CLOSE 2: CLOSE 1: END                           <055>
```

Listing 3. »Change Crill-Edi« ändert automatisch die Startadresse des Editors auf den Originalwert. Bitte mit dem Checksummer (Seite 159) eingeben.

Checksummer V3 und MSE

Diese beiden Programme sind unentbehrlich beim Abtippen unserer Listings. Sie helfen, Tippfehler vor allem bei Maschinenprogrammen zu vermeiden und sparen eine Menge Zeit.

Nobody is perfect. Jeder Computer-Fan, egal ob blutiger Anfänger oder ausgefuchster Profi, macht beim Abtippen von Programmen Tippfehler. Diese Fehler später zu finden, kann ein langwieriges Unterfangen sein.

Deshalb haben wir für Sie die Programme »Checksummer V3« und »MSE« (MaschinenSpracheEditor) entwickelt. Der Checksummer ist für Basic-Programme und der MSE für Maschinensprache-Listings zuständig.

Der Checksummer

Zuerst einmal müssen Sie das Checksummer-Programm (siehe Listing 1) abtippen. Dabei sollten Sie äußerst sorgfältig vorgehen, vor allem bei den Zahlen in den DATA-Zeilen 20 bis 30. Wenn Sie trotzdem noch einen Tippfehler gemacht haben, meldet sich das Programm später mit einem entsprechenden Hinweis. Wenn Sie fertig sind, speichern Sie das Programm auf Diskette oder Kassette.

Jetzt geht es los:

1. Starten Sie den Checksummer durch die Eingabe von »RUN« und das Drücken der RETURN-Taste.
2. Wenn die Meldung »Checksummer aktiviert...« auf dem Bildschirm erscheint, haben Sie keinen Tippfehler gemacht und der Checksummer ist nun eingeschaltet.
3. Zum Löschen des Basic-Programms geben Sie bitte »NEW« ein. Keine Angst, der Checksummer selbst wird dadurch nicht gelöscht.
4. Nun können wir den Checksummer testen. Geben Sie bitte folgende Zeile ein und drücken Sie die RETURN-Taste:

1 REM

In der linken oberen Bildschirmecke sehen Sie nun die Prüfsumme über die eben eingegebene Basic-Zeile. Sie muß <63> lauten. Dem Checksummer ist es übrigens egal, ob Sie »1 REM« oder »1REM« eintippen. Nur innerhalb von Anführungszeichen ist die richtige Anzahl an Leerzeichen wichtig. Diese Prüfsummen erscheinen (sofern Sie den Checksummer eingeschaltet haben) immer dann, wenn Sie eine Basic-Zeile eintippen und dann die RETURN-Taste drücken. In der 64'er finden Sie die Prüfsumme immer am Ende jeder Programmzeile.

```
10 PRINT"CHECKSUMMER FUER C 64"
11 PRINT:PRINT"EINEN MOMENT, BITTE ..."
12 FOR I=828 TO 864:READ A:POKE I,A:PS=PS+
   A:NEXT I
13 IF PS<>5765 THEN PRINT"TIPPFEHLER IN DE
   N ZEILEN 20 BIS 22":END
14 SYS 828:PS=0:FOR I=58464 TO 58583:READ
   A:POKE I,A:PS=PS+A:NEXT I
15 IF PS<>16147 THEN PRINT"TIPPFEHLER IN D
   EN ZEILEN 22 BIS 30":END
16 POKE 1,53:POKE 42289,96:POKE 42290,228
17 PRINT"CHECKSUMMER AKTIVIERT."
18 PRINT:PRINT" AUSSCHALTEN : POKE1,55 ODE
   R"SPC(27)"<RUN/STOP+RESTORE>"
19 PRINT:PRINT" ANSCHALTEN : POKE1,53"
20 DATA 169,0,133,254,162,1,189,93,3,133,2
   55,160,0,177,254
21 DATA 145,254,136,208,249,230,255,165,25
   5,221,95,3,208,238,202
22 DATA 16,230,96,160,224,192,0,160,2,169,
   0,170,133,254,177
23 DATA 95,240,40,201,32,208,3,200,208,245
   ,133,255,138,41,7
24 DATA 170,240,14,72,165,255,24,42,105,0,
   202,208,249,133,255
25 DATA 104,170,232,165,255,24,101,254,133
   ,254,76,111,228,192,4
26 DATA 48,219,198,214,165,214,72,162,3,16
   9,32,157,1,4,189
27 DATA 212,228,32,210,255,208,12,0,92,72,
   32,201,255,170,104
28 DATA 144,1,138,96,202,16,228,166,254,16
   9,0,32,205,189,169
29 DATA 62,32,210,255,104,133,214,32,108,2
   29,169,141,32,210,255
30 DATA 76,128,164,9,60,18,19
```

© 64'er

Listing 1. Der »Checksummer 64 V3« für Basic-Listings

```
5 PRINT CHR$(14)                              <242>
10 PRINT"{CLR}"                               <254>
20 PRINT"Q****************************W"      <130>
30 PRINT"{4DOWN,2SPACE}TEST{SPACE,BLUE,6SP
   ACE}"                                      <022>
40 PRINT"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB"      <108>
```

© 64'er

Bild 1. Die Bedeutung der Steuerzeichen wird im nachfolgenden Text erklärt

In Zeile 10 müssen Sie nach den Anführungszeichen die Tasten <SHIFT CLR/HOME> drücken und nicht die Klammern mit dem Wort CLR eingeben. In Zeile 20 drücken Sie nach den Anführungszeichen die CBM-Taste und den Buchstaben <Q>, gefolgt von mehreren SHIFT- und Stern-Tasten und zum Schluß die CBM-Taste und den Buchstaben <W>. In Zeile 30 ist es viermal die CURSOR-abwärts-Taste, gefolgt von zweimaliger Leertaste, dann <SHIFT T> und normal EST, zum Schluß noch einmal die Leertaste, die Farbtaste Blau <CTRL 7> und sechsmal die Leertaste. Zeile 40 besteht lediglich aus mehreren Grafikzeichen, die mit der CBM-Taste und <B> erzeugt werden.

CTRL steht für Control-Taste, so bedeutet {CTRL+A}, daß Sie die Control-Taste und die Taste »A« drücken müssen. Im folgenden steht:

{DOWN}	Taste neben rechtem Shift, Cursor unten	{SPACE}	Leertaste	{RVSON}	Control-Taste & 9
{UP}	Shift-Taste & Taste neben rechtem Shift; Cursor hoch	{SHIFT-Space}	Shift-Taste & Leertaste	{RVOFF}	Control-Taste & 0
{CLR}	Shift-Taste & 2. Taste ganz rechts oben	{F1} bis {F8}	Funktionstasten	{ORANGE}	Commodore-Taste & 1
{INST}	Shift-Taste & Taste ganz rechts oben	{RETURN}	Return-Taste	{BROWN}	Commodore-Taste & 2
{HOME}	2. Taste von ganz rechts oben	{BLACK}	Control-Taste & 1	{LIG.RED}	Commodore-Taste & 3
{DEL}	Taste ganz rechts oben	{WHITE}	Control-Taste & 2	{GREY 1}	Commodore-Taste & 4
{RIGHT}	Taste ganz rechts unten	{RED}	Control-Taste & 3	{GREY 2}	Commodore-Taste & 5
{LEFT}	Shift-Taste & Taste unten rechts	{CYAN}	Control-Taste & 4	{LIG.GREEN}	Commodore-Taste & 6
		{PURPLE}	Control-Taste & 5	{LIG.BLUE}	Commodore-Taste & 7
		{GREEN}	Control-Taste & 6	{GREY 3}	Commodore-Taste & 8
		{BLUE}	Control-Taste & 7		
		{YELLOW}	Control-Taste & 8		

Tabelle 1. Die Steuerbefehle in den Listings

Diese Zahlen dürfen Sie NICHT mit abtippen.

Als Beispiel sehen Sie Bild 1. Am rechten Rand jeder Spalte sehen Sie die Prüfsummen in eckigen Klammern.

Damit sind wir beim zweiten wichtigen Punkt: Sehen Sie sich die Zeile 240 von Listing 2 genauer an. Nach dem ersten Anführungszeichen nach dem PRINT-Befehl sehen Sie eine geschweifte Klammer { }. Immer, wenn Sie in einem unserer Listings diese Klammern sehen, dürfen Sie das, was innerhalb der Klammern steht, nicht eintippen. Sie müssen die entsprechende Taste drücken. Beispiel:
10 PRINT "{CLR}"
bedeutet: Nach dem Anführungszeichen die »Bildschirmlöschen«-Taste drücken (<SHIFT CLR/HOME>). In Tabelle 1 sehen Sie eine Zusammenfassung aller möglichen Steuertasten mit dem entsprechenden Klartext.

Weiterhin sehen Sie in Bild 1 (Bedeutung der Steuerzeichen) in Zeile 30 ein unterstrichenes »T« nach der Klammer. Das bedeutet, daß Sie ein »T« zusammen mit der SHIFT-Taste drücken müssen, also <SHIFT T>. Wenn ein Zeichen »überstrichen« ist, müssen Sie dieses zusammen mit der CBM-Taste eingeben. Die CBM-Taste befindet sich ganz links unten auf der Tastatur und hat die Aufschrift »C=«.

```
100 REM DIESES PROGRAMM ERZEUGT DEN            <210>
110 REM MSE V1.1 AUF DISKETTE.                 <039>
120 REM BESITZER EINER DATASETTE               <178>
130 REM MUESSEN DIE '8' AM ENDE VON            <145>
140 REM ZEILE 343 IN EINE '1' AENDERN!         <176>
150 REM                                        <212>
230 IF PEEK(44)<>32 THEN PRINT"{CLR}SIE HA
    BEN VERGESSEN, DIE POKES EINZUGE- BEN!
    ":END                                      <050>
240 PRINT"{CLR}";:DIM H(75):FOR I=0 TO 9       <042>
250 H(48+I)=I:H(65+I)=I+10:NEXT:Z=1000         <136>
260 FOR I=2048 TO 3755 STEP 20:PRINT"{HOME
    }ICH LESE ZEILE:"Z                         <253>
261 FOR N=0 TO 19:READ A$:IF LEN(A$)<>2 TH
    EN 900                                     <062>
262 IF PEEK(63)+PEEK(64)*256<>Z THEN 800       <011>
270 H=ASC(LEFT$(A$,1)):L=ASC(RIGHT$(A$,1))     <199>
280 D=H(H)*16+H(L):S=S+D:POKE I+N,D            <165>
290 NEXT:READ V:IF S<>V THEN 900               <139>
300 S=0:Z=Z+1:NEXT:R=PEEK(2111):H=PEEK(210
    6)                                         <126>
301 POKE 53280,R:POKE 53281,H:POKE 646,R:P
    RINT"{CLR}DIE DATA-ZEILEN SIND FEHLERF
    REI!"                                      <080>
302 PRINT"SIE KOENNEN NUN DIE FARBEN DES M
    SE"                                        <209>
303 PRINT"EINSTELLEN.":PRINT"{2DOWN,SPACE,
    RVSON}DRUECKEN SIE <1>, <2> ODER <9>       <205>
304 PRINT"{DOWN,2SPACE}<1> - RAHMEN-/SCHRI
    FTFARBE                                    <013>
305 PRINT"{2SPACE}<2> - HINTERGRUNDFARBE       <233>
306 PRINT"{DOWN,2SPACE}<9> - FARBEN UEBERN
    EHMEN                                      <158>
307 PRINT"{2DOWN}FARBE <1>:"R:PRINT"FARBE
    <2>:"H                                     <066>
308 GET A:IF A=0 THEN 308                      <210>
309 IF A=1 THEN R=(R+1)AND 15                  <098>
310 IF A=2 THEN H=(H+1)AND 15                  <086>
311 IF A=9 THEN 340                            <217>
312 GOTO 301                                   <034>
340 POKE 2106,H:POKE 2111,R                    <153>
342 POKE 631,19:POKE 632,13:POKE 198,2         <135>
343 PRINT"{CLR}SAVE"CHR$(34)"MSE V1.1"CHR$
    (34)",8                                    <091>
344 POKE 43,1:POKE 44,8:POKE 45,172:POKE 4
    6,14:END                                   <140>
800 PRINT"{CLR,RVSON}SIE HABEN ZEILE"Z"{LE
    FT,SPACE}VERGESSEN:":A=PEEK(646)AND 15     <124>
810 POKE 646,PEEK(53281)AND 15:PRINT"LIST"
    Z-2"-"Z+2:POKE 646,A                       <224>
820 GOTO 920                                   <082>
900 PRINT"{CLR,RVSON}SIE HABEN EINEN TIPPF
    EHLER GEMACHT:":A=PEEK(646)AND 15          <154>
910 POKE 646,PEEK(53281)AND 15:PRINT"LIST"
    Z:POKE 646,A                               <173>
920 POKE 631,19:POKE 632,17:POKE 633,13:PO
    KE 198,3:END                               <126>
1000 DATA 00,0B,08,0A,00,9E,32,30,36,31,00
     ,00,00,A2,08,A9,36,85,A4,A9, 1247         <119>
1001 DATA 08,85,A5,A9,00,85,A6,A9,B0,85,A7
     ,A0,00,B1,A4,91,A6,C8,D0,F9, 2888         <054>
1002 DATA E6,A5,E6,A7,CA,D0,F2,A9,36,85,01
     ,4C,00,B0,20,D1,B1,A9,00,8D, 2781         <096>
1003 DATA 21,D0,A9,0F,8D,20,D0,8D,86,02,A0
     ,B3,A9,74,20,FF,B1,A0,B3,A9, 2679         <089>
1004 DATA B9,20,FF,B1,A0,00,20,CF,FF,99,01
     ,02,C8,C9,0D,D0,F5,88,F0,D2, 2912         <217>
1005 DATA C0,11,90,02,A0,10,8C,00,02,20,EA
     ,B1,A0,B3,A9,CF,20,FF,B1,20, 2327         <045>
1006 DATA 8E,B4,85,FC,85,62,20,8E,B4,85,FB
     ,85,61,20,A7,B4,D0,20,A0,B3, 2864         <199>
1007 DATA A9,E5,20,FF,B1,20,8E,B4,85,60,20
     ,8E,B4,85,5F,20,A7,B4,D0,0A, 2624         <091>
1008 DATA A5,61,C5,5F,A5,62,E5,60,90,06,20
     ,43,B3,4C,3A,B0,A9,AA,A0,00, 2379         <167>
1009 DATA EA,EA,E6,FB,D0,02,E6,FC,20,3F,B2
     ,90,EF,4C,FB,B4,A2,02,86,58, 3190         <041>
1010 DATA A9,A6,A0,9D,20,F2,B1,20,E4,FF,F0
     ,FB,C9,30,90,0C,C9,47,B0,08, 2970         <231>
1011 DATA C9,3A,90,0B,C9,41,B0,07,C9,14,D0
     ,0F,4C,0B,B1,20,D2,FF,A6,58, 2322         <121>
1012 DATA 95,F7,C6,58,D0,D2,60,AE,8D,02,F0
     ,26,C9,0C,D0,03,4C,0B,B6,C9, 2685         <057>
1013 DATA 13,D0,03,4C,8B,B5,C9,0D,D0,03,4C
     ,BA,B4,C9,10,D0,03,4C,68,B5, 2282         <225>
1014 DATA C9,0E,D0,06,20,5F,B4,4C,64,B1,4C
     ,92,B0,A5,F9,20,02,B1,0A,0A, 2132         <208>
1015 DATA 0A,0A,85,F9,A5,F8,20,02,B1,05,F9
     ,60,C9,3A,90,02,69,08,29,0F, 1950         <092>
1016 DATA 60,A6,59,E0,08,90,1F,A6,58,E0,02
     ,B0,06,20,D2,FF,4C,8E,B0,C6, 2509         <188>
1017 DATA 59,A0,14,A9,92,20,F2,B1,CA,D0,FA
     ,84,57,68,68,4C,8B,B1,A6,D3, 2891         <197>
1018 DATA E0,08,B0,03,4C,92,B0,20,D2,FF,A6
     ,58,E0,02,90,09,C6,59,20,D2, 2468         <049>
1019 DATA FF,C6,58,D0,F9,4C,8E,B0,48,4A,4A
     ,4A,4A,20,59,B1,68,29,0F,C9, 2419         <035>
1020 DATA 0A,90,02,69,06,69,30,4C,D2,FF,A2
     ,FC,9A,20,D1,B1,20,48,B2,20, 2261         <073>
1021 DATA EA,B1,20,9F,B2,A5,FC,20,4E,B1,A5
     ,FB,20,4E,B1,20,ED,B1,A9,3A, 2860         <148>
1022 DATA A0,20,20,F2,B1,A9,00,85,59,20,8E
     ,B0,20,ED,B1,A4,59,20,EF,B0, 2530         <233>
1023 DATA 91,FB,C8,84,59,C0,08,90,EC,20,10
     ,B2,A9,12,20,D2,FF,20,8E,B0, 2657         <105>
1024 DATA 20,EF,B0,C5,FF,F0,0D,20,43,B3,A9
     ,14,A0,14,20,F2,B1,4C,A2,B1, 2665         <034>
1025 DATA A9,92,20,D2,FF,20,33,B2,20,E0,B2
     ,20,3F,B2,90,9F,4C,8B,B5,A9, 2648         <123>
1026 DATA 93,20,D2,FF,A2,00,A9,03,9D,00,D8
     ,9D,00,D9,9D,00,DA,9D,00,DB, 2476         <237>
1027 DATA E8,D0,EF,60,A9,0D,2C,A9,20,4C,D2
     ,FF,20,D2,FF,98,4C,D2,FF,20, 2965         <160>
1028 DATA E4,FF,F0,FB,60,84,5D,85,5C,A0,00
     ,B1,5C,F0,06,20,D2,FF,C8,D0, 3100         <077>
1029 DATA F6,60,A5,FB,85,5A,A0,00,84,5B,B1
     ,FB,18,65,5A,85,5A,90,02,E6, 2606         <156>
1030 DATA 5B,06,5A,26,5B,C8,C0,08,90,EC,A5
     ,5A,65,5B,85,FF,60,18,A5,FB, 2467         <219>
1031 DATA 69,08,85,FB,90,02,E6,FC,60,A5,FB
     ,C5,5F,A5,FC,E5,60,60,A0,B3, 3106         <183>
1032 DATA A9,FB,20,FF,B1,A0,01,B9,00,02,20
     ,D2,FF,CC,00,02,C8,90,F4,A9, 2692         <098>
1033 DATA 14,ED,00,02,AA,20,ED,B1,CA,D0,FA
     ,A5,62,20,4E,B1,A5,61,20,4E, 2457         <060>
1034 DATA B1,20,ED,B1,A5,60,20,4E,B1,A5,5F
     ,20,4E,B1,EA,EA,EA,EA,EA,EA, 3122         <190>
1035 DATA EA,EA,24,5E,10,01,60,A9,12,20,D2
     ,FF,A2,28,20,ED,B1,CA,D0,FA, 2703         <087>
1036 DATA A9,92,4C,D2,FF,A5,D6,C9,16,B0,01
     ,60,A9,A0,85,A4,A9,78,85,A6, 2945         <204>
1037 DATA A9,04,85,A5,85,A7,A2,13,A0,27,B1
     ,A4,91,A6,88,10,F9,CA,F0,19, 2671         <208>
1038 DATA 18,A5,A4,69,28,85,A4,90,02,E6,A5
     ,18,A5,A6,69,28,85,A6,90,E0, 2503         <251>
1039 DATA E6,A7,4C,B6,B2,A9,91,4C,D2,FF,A9
     ,0F,8D,18,D4,A9,00,8D,05,D4, 2776         <000>
1040 DATA A9,F7,8D,06,D4,A9,11,8D,04,D4,A9
     ,32,8D,01,D4,A9,00,8D,00,D4, 2413         <126>
1041 DATA A0,80,20,09,B3,A9,10,8D,04,D4,60
     ,A2,FF,CA,D0,FD,88,D0,F8,60, 2914         <240>
1042 DATA A9,0F,8D,18,D4,A9,2D,8D,05,D4,A9
     ,A5,8D,06,D4,A9,21,8D,04,D4, 2385         <119>
1043 DATA A9,07,8D,01,D4,A9,05,8D,00,D4,A0
```

```
     ,FF,2Ø,Ø9,B3,A9,2Ø,8D,Ø4,D4, 225Ø      <Ø78>
1Ø44 DATA A9,ØØ,8D,Ø1,D4,8D,ØØ,D4,6Ø,38,2Ø
     ,FØ,FF,8A,48,98,48,18,AØ,Ø6, 2179      <175>
1Ø45 DATA A2,18,2Ø,FØ,FF,AØ,B4,A9,ØA,2Ø,FF
     ,B1,2Ø,12,B3,2Ø,E4,FF,FØ,FB, 2931      <Ø93>
1Ø46 DATA A2,1D,A9,14,2Ø,D2,FF,CA,DØ,FA,68
     ,A8,68,AA,18,4C,FØ,FF,ØD,ØD, 27Ø4      <Ø88>
1Ø47 DATA ØD,2Ø,2Ø,2Ø,2Ø,2Ø,2Ø,2Ø,4D,41,53
     ,43,48,49,4E,45,4E,53,5Ø,52, 1144      <216>
1Ø48 DATA 41,43,48,45,2Ø,2D,2Ø,45,44,49,54
     ,4F,52,2Ø,ØD,ØD,2Ø,2Ø,2Ø,2Ø, 1Ø23      <Ø38>
1Ø49 DATA 2Ø,2Ø,2Ø,2Ø,56,4F,4E,2Ø,4E,2E,4D
     ,41,4E,4E,2Ø,26,2Ø,44,2E,57, 1128      <2Ø6>
1Ø5Ø DATA 45,49,4E,45,43,4B,00,ØD,ØD,ØD,2Ø
     ,2Ø,2Ø,5Ø,52,4F,47,52,41,4D, 11Ø2      <117>
1Ø51 DATA 4D,4E,41,4D,45,2Ø,3A,2Ø,ØØ,ØD,ØD
     ,2Ø,2Ø,2Ø,53,54,41,52,54,41, 1Ø73      <Ø95>
1Ø52 DATA 44,52,45,53,53,45,2Ø,3A,2Ø,24,ØØ
     ,ØD,ØD,2Ø,2Ø,2Ø,45,4E,44,41, 1Ø14      <129>
1Ø53 DATA 44,52,45,53,53,45,2Ø,2Ø,2Ø,3A,2Ø
     ,24,ØØ,92,Ø1,Ø1,5Ø,52,4F,47, 1138      <228>
1Ø54 DATA 52,41,4D,4D,2Ø,3A,2Ø,ØØ,12,2Ø,2Ø
     ,2A,2A,2A,2Ø,46,41,4C,53,43, 1Ø24      <Ø27>
1Ø55 DATA 48,45,2Ø,45,49,4E,47,41,42,45,2Ø
     ,2A,2A,2A,2Ø,2Ø,92,ØØ,ØD,ØD, 1Ø58      <Ø98>
1Ø56 DATA 2A,2A,2A,2Ø,45,4E,44,45,2Ø,2A,2A
     ,2A,ØØ,13,Ø1,2Ø,2Ø,12,44,92, 916       <153>
1Ø57 DATA 49,53,4B,2Ø,4F,44,45,52,2Ø,12,54
     ,92,41,5Ø,45,ØD,ØØ,13,2Ø,2Ø, 1151      <Ø35>
1Ø58 DATA 49,2F,4F,2Ø,2D,2Ø,46,45,48,4C,45
     ,52,ØØ,2Ø,D1,B1,2Ø,48,B2,AØ, 16Ø6      <Ø12>
1Ø59 DATA B3,A9,CF,2Ø,FF,B1,2Ø,8E,B4,85,FC
     ,2Ø,8E,B4,85,FB,C5,61,A5,FC, 32Ø7      <251>
1Ø6Ø DATA E5,62,9Ø,23,A5,FB,C5,5F,A5,FC,E5
     ,6Ø,BØ,19,2Ø,A7,B4,DØ,14,6Ø, 286Ø      <112>
1Ø61 DATA 2Ø,A7,B4,FØ,ØC,85,F9,2Ø,A7,B4,FØ
     ,Ø5,85,F8,4C,EF,BØ,68,68,2Ø, 2749      <Ø88>
1Ø62 DATA 43,B3,4C,5F,B4,2Ø,CF,FF,C9,4C,DØ
     ,Ø9,2Ø,D1,B1,2Ø,48,B2,4C,ØB, 2372      <Ø46>
1Ø63 DATA B6,C9,ØD,6Ø,A9,ØØ,85,5E,2Ø,5F,B4
     ,2Ø,EA,B1,2Ø,ØD,B5,24,5E,3Ø, 2Ø42      <12Ø>
1Ø64 DATA Ø5,2Ø,E4,FF,FØ,FB,2Ø,E1,FF,FØ,26
     ,2Ø,9F,B2,24,5E,1Ø,Ø9,2Ø,4E, 2435      <198>
1Ø65 DATA B5,2Ø,ØD,B5,2Ø,6Ø,B5,2Ø,33,B2,2Ø
     ,3F,B2,9Ø,D7,AØ,B4,A9,28,2Ø, 219Ø      <2Ø7>
1Ø66 DATA FF,B1,2Ø,E4,FF,C9,ØD,DØ,F9,A9,ØØ
     ,85,5E,A5,61,85,FB,A5,62,85, 3Ø56      <24Ø>
1Ø67 DATA FC,2Ø,EØ,B2,4C,64,B1,A5,FC,2Ø,4E
     ,B1,A5,FB,85,FF,2Ø,4E,B1,A9, 3ØØ3      <221>
1Ø68 DATA 2Ø,AØ,3A,2Ø,F2,B1,AØ,ØØ,2Ø,ED,B1
     ,B1,FB,2Ø,4E,B1,C8,CØ,Ø8,9Ø, 2566      <Ø7Ø>
1Ø69 DATA F3,2Ø,ED,B1,24,5E,3Ø,Ø3,A9,12,2C
     ,A9,2Ø,2Ø,D2,FF,2Ø,1Ø,B2,A5, 219Ø      <Ø59>
1Ø7Ø DATA FF,2Ø,4E,B1,A9,92,2Ø,D2,FF,4C,EA
     ,B1,A9,FF,85,B8,85,B9,A9,Ø4, 3Ø73      <Ø29>
1Ø71 DATA 85,BA,2Ø,CØ,FF,A2,FF,4C,C9,FF,2Ø
     ,CC,FF,A9,FF,4C,C3,FF,2Ø,5F, 3315      <189>
1Ø72 DATA B4,A9,8Ø,85,5E,2Ø,4E,B5,2Ø,48,B2
     ,A2,24,A9,2D,2Ø,D2,FF,CA,DØ, 2596      <111>
1Ø73 DATA FA,2Ø,EA,B1,2Ø,EA,B1,2Ø,6Ø,B5,4C
     ,C1,B4,2Ø,B8,B5,A6,5F,A4,6Ø, 2812      <Ø15>
1Ø74 DATA A9,61,2Ø,D8,FF,BØ,ØA,2Ø,B7,FF,29
     ,BF,DØ,Ø3,4C,FB,B4,A9,Ø1,2Ø, 2577      <2Ø1>
1Ø75 DATA C3,FF,2Ø,68,B6,AØ,B4,A9,4F,2Ø,FF
     ,B1,2Ø,F9,B1,4C,FB,B4,2Ø,68, 2921      <237>
1Ø76 DATA B6,A9,37,AØ,B4,2Ø,FF,B1,2Ø,F9,B1
     ,A2,Ø8,C9,44,FØ,Ø6,A2,Ø1,C9, 2717      <213>
1Ø77 DATA 54,DØ,F1,A9,Ø1,A8,2Ø,BA,FF,AØ,ØØ
     ,EØ,Ø1,FØ,1A,A9,4Ø,8D,2Ø,Ø2, 24Ø3      <1Ø1>
1Ø78 DATA A9,3A,8D,21,Ø2,B9,Ø1,Ø2,99,22,Ø2
     ,C8,CC,ØØ,Ø2,9Ø,F4,C8,C8,DØ, 2182      <127>
1Ø79 DATA ØC,B9,Ø1,Ø2,99,2Ø,Ø2,C8,CC,ØØ,Ø2
     ,DØ,F4,98,A2,2Ø,AØ,Ø2,4C,BD, 2Ø18      <Ø25>
1Ø8Ø DATA FF,2Ø,B8,B5,A5,BA,C9,Ø8,9Ø,33,A6
     ,B9,86,57,A9,Ø1,2Ø,C3,FF,A9, 28ØØ      <Ø22>
1Ø81 DATA 6Ø,85,B9,2Ø,CØ,FF,BØ,28,A5,BA,2Ø
     ,B4,FF,A5,B9,2Ø,96,FF,2Ø,A5, 2911      <Ø53>
1Ø82 DATA FF,85,61,A5,9Ø,4A,4A,BØ,13,2Ø,A5
     ,FF,85,62,2Ø,AB,FF,A5,57,85, 2663      <214>
1Ø83 DATA B9,A9,ØØ,2Ø,D5,FF,9Ø,Ø3,4C,A3,B5
     ,86,5F,84,6Ø,A5,BA,C9,Ø1,DØ, 2639      <131>
1Ø84 DATA ØA,AD,3D,Ø3,85,61,AD,3E,Ø3,85,62
     ,4C,FB,B4,A9,13,2Ø,D2,FF,A2, 23ØØ      <12Ø>
1Ø85 DATA 1C,2Ø,ED,B1,CA,DØ,FA,6Ø,ØØ,ØØ,ØØ
     ,ØØ,ØØ,ØØ,ØØ,ØØ,ØØ,ØØ,ØØ,ØØ, 123Ø      <143>
```

 Listing 2. Der MSE-Lader

Der MSE

Der MSE dient zur Eingabe von Maschinensprache-Programmen. Als erstes müssen Sie den sogenannten »MSE-Lader« (Listing 2) abtippen. Dieser erzeugt erst das eigentliche MSE-Programm auf Diskette oder Kassette.

Wichtig: Vor dem Eintippen des MSE-Laders müssen Sie unbedingt ein paar Befehle eingeben (ohne Basic-Zeilennummer): POKE 44,32 : POKE 8192,0 : NEW

Jetzt können Sie beginnen, das Listing 2 abzutippen. Der MSE-Lader erkennt zwar, wenn Sie beim Eintippen der DATA-Zeilen einen Fehler gemacht haben, aber wenn Sie ganz sicher gehen möchten, sollten Sie den Checksummer vor dem Eintippen aktivieren. Die Prüfsummen für den MSE-Lader finden Sie am Ende der jeweiligen Programmzeilen.

Wenn Sie das Listing 2 nicht auf einmal abtippen möchten, müssen Sie vor jedem neuen Laden des Programms unbedingt die oben genannte POKE-Zeile eingeben!

Wenn Sie alles richtig gemacht haben und das Programm fehlerfrei abgetippt wurde, speichert es sich nach dem Starten selbst auf Diskette oder Kassette unter dem Namen »MSE V1.0«. Dieses fertige MSE-Programm laden Sie dann bei Bedarf wie ein normales Basic-Programm und starten es mit »RUN«.

So arbeitet man mit dem MSE

Als erstes möchte der MSE den Namen des zu bearbeitenden Programms wissen. Dieser steht in der ersten Zeile unserer MSE-Listings. Dann müssen Sie die Start- und Endadresse des Programms eingeben. Dies sind die letzten beiden vierstelligen Hexadezimalzahlen in der ersten Zeile unserer Listings.

Wenn Sie ein Programm von Diskette oder Kassette laden wollen, um an einer bestimmten Stelle weiterzutippen oder noch eine Korrektur vorzunehmen, geben Sie auf die Frage nach der Startadresse ein »L« ein. Danach müssen Sie <D> oder <T> drücken, je nachdem, ob Sie von Diskette oder Kassette (»tape«) laden möchten. Wenn das Programm unter diesem Namen nicht auf der Diskette vorhanden ist oder ein sonstiger Ladefehler vorlag, meldet sich der MSE mit »I/O-ERROR«. In diesem Fall drücken Sie <RUN/STOP RESTORE> und geben einfach noch einmal »RUN« ein.

Beim Abtippen geben Sie nach und nach die abgedruckten Buchstaben und Zahlen des jeweiligen Listings ohne die Freiräume dazwischen ein. Wenn Sie in einer Zeile einen Tippfehler gemacht haben, meldet sich der MSE sofort mit einem Brummton und der Meldung »EINGABEFEHLER«. Nach einem Druck auf die RETURN-Taste können Sie mit der DEL-Taste den Fehler korrigieren. Wenn Sie das gewünschte Programm vollständig eingegeben haben, speichert es der MSE automatisch auf Diskette oder Kassette.

Bei längeren Listings ist es unwahrscheinlich, daß Sie das komplette Programm auf einmal eingeben. Sie können Ihre bisherige Tipparbeit jederzeit durch <CTRL S> auf Diskette oder Kassette speichern und Ihr Werk später fortsetzen. Sie sollten sich dann allerdings im Heft markieren, wie weit Sie beim Abtippen gekommen sind! Später geben Sie dann nach dem Laden des ersten Programmteils <CTRL N> ein und auf die dann folgende Frage nach der Startadresse die Zeilennummer (Adresse), bei der Sie aufgehört haben zu tippen.

<CTRL M> erlaubt Ihnen jederzeit, Ihr Werk listen zu lassen. Durch <SPACE> können Sie weiterlisten lassen und durch <RUN/STOP> das Listen abbrechen.

Wenn Sie einen Drucker besitzen, können Sie das Programm auch mit <CTRL P> ausdrucken. Mit <CTRL L> wird das Programm noch einmal neu in Ihren C 64 geladen.

(F. Lonczewski/N. Mann/D. Weineck/ef)

SONDERHEFT 44

VORSCHAU





Das Sonderheft 44 ist für alle C128-Fans eine Fundgrube nützlicher Anwendungsprogramme:

Mit »Dispo 128« verwalten Sie komfortabel Ihre Diskettensammlung. Auch die 1581 wird vom Programm automatisch erkannt.

Strukturiertes Programmieren wird mit »Flowchart« zum Kinderspiel. Entwerfen Sie das Flußdiagramm, das dazugehörige Basic-Programm erstellt »Flowchart« automatisch.

Ein Zeichenprogramm der Spitzenklasse ist »Gredi«. Der Clou: Der 80-Zeichen-Bildschirm wird voll ausgenutzt.

Rasend schnell erlaubt »Floppy Support« umfangreiche Diskettenmanipulationen für die 1570/71.

Das Sonderheft 44 liegt ab dem 28.7.1989 an Ihrem Kiosk.

Impressum

__Herausgeber:__ Carl-Franz von Quadt, Otmar Weber

__Chefredakteur:__ Hans-Günther Beer
__Stellv. Chefredakteur:__ Gottfried Knechtel - verantwortlich für den redaktionellen Teil
__Chef vom Dienst:__ Susanne Kirmaier
__Redaktion:__ Ralf Sablowski, Elmar Friebe, Klaus Sonnenleiter, Andreas Greil
__Redaktionsassistenz:__ Brigitte Bobenstetter, Helga Weber (202), Sylvia Derenthal
__Hotline:__ Monika Welzel (640)
__Mitarbeiter der Redaktion:__ Dr. Rudolf Egg, Nikolaus Heusler, Nikolaus Huber, Stefan Seidler

Alle Artikel sind mit dem Kennzeichen des Redakteurs (kn = Gottfried Knechtel, rs = Ralf Sablowski, ef = Elmar Friebe, so = Klaus Sonnenleiter, ag = Andreas Greil und/oder mit dem Namen des Autors/Mitarbeiters gekennzeichnet

__Art-director:__ Friedemann Porscha
__Layout:__ Erich Schulze (Cheflayouter), Marian Schwarz, Johanna Schneider
__Fotografie:__ Sabine Tennstaedt, Ilona Wiewiorra, Roland Müller
__Titelgestaltung:__ Friedemann Porscha, Erich Schulze
__Spritzgrafik:__ Norbert Raab
__Computergrafik:__ Werner Nienstedt (Titelmotiv)

__Auslandsrepräsentation:__
Schweiz: Markt&Technik Vertriebs AG, Kollerstr. 3, CH-6300 Zug, Tel. 042-415656, Telex: 862329 mut ch
USA: M&T Publishing Inc.; 501 Galveston Drive Redwood City, CA 94063, Telefon: (415) 366-3600, Telex 752-351
Österreich: Markt&Technik Ges. mbH Große Neugasse 28, A 1040-Wien, Tel. 0222/5871393, Telex: 047-132532

__Manuskripteinsendungen:__ Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Sie müssen frei sein von Rechten Dritter. Sollten sie auch an anderer Stelle zur Veröffentlichung oder gewerblichen Nutzung angeboten worden sein, muß dies angegeben werden. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck in von der Markt&Technik Verlag AG herausgegebenen Publikationen und zur Vervielfältigung der Programmlistings auf Datenträger. Mit der Einsendung von Bauanleitungen gibt der Einsender die Zustimmung zum Abdruck in von Markt&Technik Verlag AG verlegten Publikationen und dazu, daß Markt&Technik Verlag AG Geräte und Bauteile nach der Bauanleitung herstellen läßt und vertreibt oder durch Dritte vertreiben läßt. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

__Produktionsleiter:__ Klaus Buck (180); Wolfgang Meyer (stellv.) (887)

__Anzeigenleitung:__ Phillip Schiede (399) - verantwortlich für Anzeigen

__Anzeigenformate:__ ¼ Seite ist 266 Millimeter hoch und 185 Millimeter breit (2 Spalten à 86 Millimeter oder 4 Spalten à 43 Millimeter). Vollformat 297x210 Millimeter.

__Anzeigenpreise:__ Es gilt die Anzeigenpreisliste vom 5. Januar 1988. ¼-Seite sw: DM 5400,-. Farbzuschlag: erste und zweite Farbe aus der Europa-Skala je DM 1000,-. Vierfarbzuschlag DM 2800,-. Plazierung innerhalb der redaktionellen Beiträge. Mindestgröße ¼-Seite.

__Anzeigenverwaltung und Disposition:__ Lisa Landthaler (233)

__Anzeigen-Auslandsvertretung: England:__ F. A. Smyth&Associates Limited, 23a, Aylmer Parade, London, N2 OPQ. Telefon:0044/1/3405058, Telefax: 0044/1/3419602
__Taiwan:__ Third Wave Publishing Corp., 1-4 Fl. 977 Min Shen E. Road, Taipei 10581, Taiwan, R.O.C., Tel. 00886/2/7630052, Telefax: 00886/2/7658767, Telex: 078529335

__Vertriebsleiter:__ Helmut Grünfeldt (189)

__Verkaufsleiter Abonnement:__ Benno Gaab (740)

__Verkaufsleiter Einzelhandel:__ Robert Riesinger (364)

__Vertrieb Handelsauflage:__ Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Hauptstätter Straße 96, 7000 Stuttgart 1

__Bezugsmöglichkeiten:__ Leser-Service: Telefon (089) 4613-249. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen.

__Preis:__ Das Einzelheft kostet DM 14,-

__Druck:__ SOV Graphische Betriebe, Laubanger 23, 8600 Bamberg

__Sonderdruck-Dienst:__ Alle in dieser Ausgabe erschienenen Beiträge sind in Form von Sonderdrucken zu erhalten. Anfragen an Reinhard Jarczok, Tel. 089/4613-185, Fax 4613-776.

__Redaktionsdirektor:__ Michael M. Pauly

__Vorstand:__ Otmar Weber (Vors.), Bernd Balzer

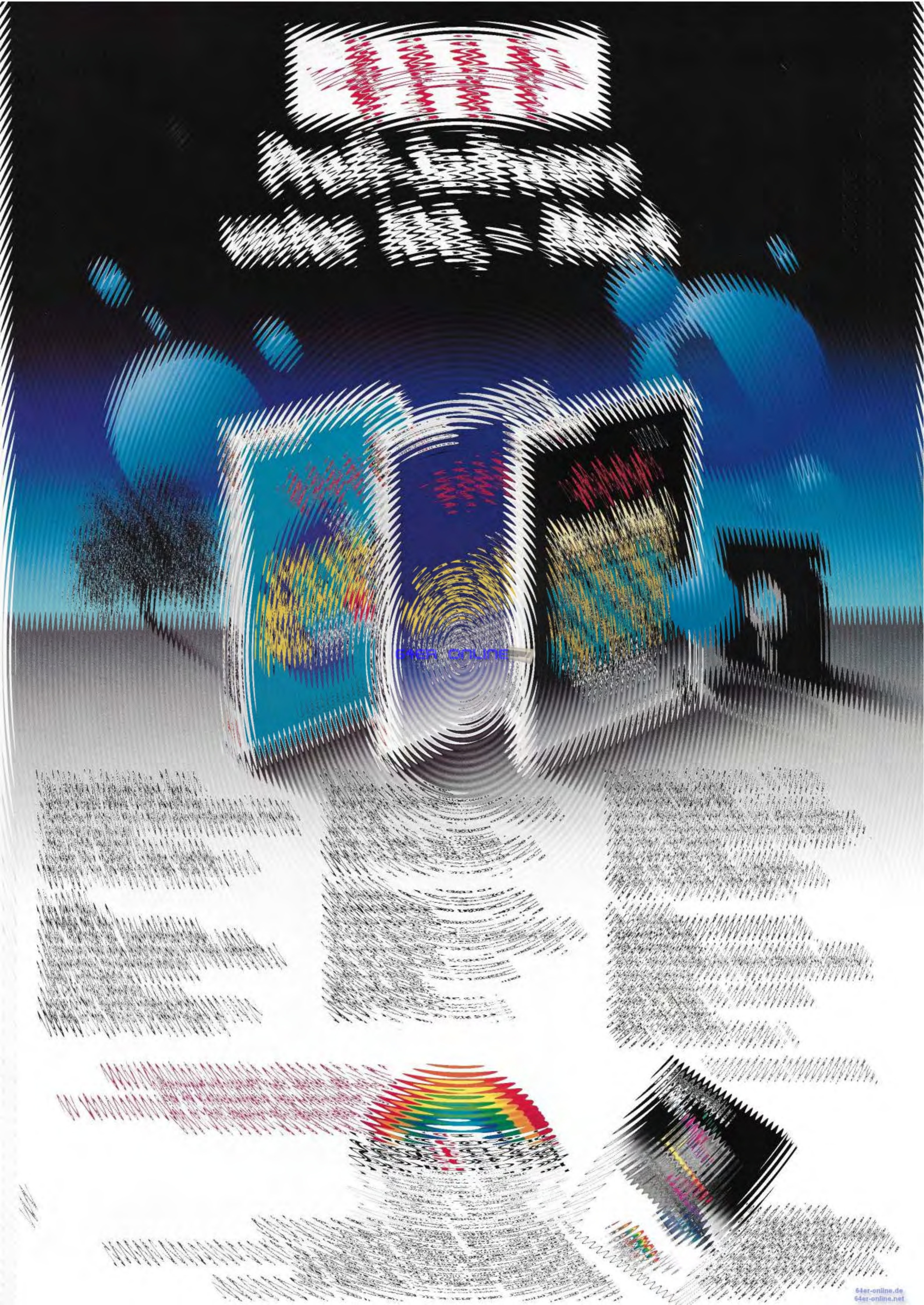
__Leiter Unternehmensbereich »Populäre Computerzeitschriften«:__ Eduard Heilmayr, Werner Pest

__Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:__ Markt&Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon (089) 4613-0, Telex 5-22052

__ISSN 0931-8933__

__Telefon-Durchwahl im Verlag:__ Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089/4613 und dann die Nummer, die in den Klammern hinter dem jeweiligen Namen angegeben ist.




64ER ONLINE


GEOS
POWER PACK